Summer 1983

Personal SOFTWARE

An Argus Specialist Publication

£1.95

Programs galore for your...

VIC-20, PET, 8000 machines, and Commodore 64.

- Ingenious utilities to run
- Educational software to test the family
- Data base for better business
- Great games to play

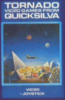
CBM
commodore
computer
64
commodore VIC-20
commodore
8032-SK

BRITAIN'S BEST BUY FOR THE COMMODORE USER

VIC20 GAMES FROM QUICKSILVA

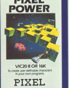
QUICKSILVA are proud to introduce
Skyhawk written by Chartec
for the VIC20 personal computer,
many more amazing games to come!

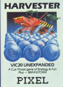
SKYHAWK: Features Multi-colour, Hi-res 3-D Effect Graphics, Realistic Scrolling Landscape, Aircraft Landing and Refueling, Radar and Aircraft Status Displays, High Score Save, Full Sound Effects, Varying Formations of Attacking Aircraft, Extra Life at 3000 pts, Fuel Low Warning, Realistic Explosions. For the VIC20 + 3K or 8K expansion + Joystick.

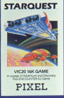

TORNADO
VIC20 GAMES FROM
QUICKSILVA
VIC20
+JOYSTICK

TORNADO: In the midst of the Colony Wars ships are attacking you from the air, you have three types of ground base to bomb. Features: Full Colour, Hi-Res Graphics, Sound, Increasing Speed, High Score, Progressive Difficulty, Random Landscape, Explosions. By Chartec for the unexpanded VIC20 + Joystick.

PIXEL GAMES


PIXEL
POWER
VIC20 8 OR 16K
To create user-definable characters
in your own programs.
PIXEL


HARVESTER
VIC20 UNEXPANDED
A Cut-Throat game of Strategy & Fun
Plus — BRAINSTORM
PIXEL


STARQUEST
VIC20 16K GAME
A voyage of Adventure and Discovery
Plus ENCOUNTER-IQ Game
PIXEL

HARVESTER & BRAINSTORM

A cut-throat strategy game to reap valuable boosterspice around the planet Delta. Hi-Res Graphics and lots of fun for two to four players. For the unexpanded VIC20.

STARQUEST/ ENCOUNTER

A voyage of discovery and adventure in the cosmos. With the help of your onboard computer you seek a habitable planet amidst the perils of deep space. For the VIC20 + 16K RAM.

PIXEL POWER

A graphics workshop packed with useful features such as Create, Amend, Save and View Set.
For the VIC20 with 8K or more added RAM.

SUBSPACE STRIKER & ZOR

It comes from out of nowhere and then vanishes back into the ether. With your deadly antimat torpedoes, you unleash havoc in the Federation's Spacelanes.
For the VIC20 + 16K RAM.

TRADER

A trilogy of 16K programs that combine to give an epic 48K graphic adventure. As a galactic trader, you deal with some very bizarre customers indeed. Will you live to tell the tale? Supplied in a box with extensive instruction booklet.

TRADER
PIXEL

For the VIC20 + 16K RAM.

Tornado	£5.95	☐
Skyhawk	£7.95	☐
Trader	£14.95	☐
Subspace Striker + Zor	£7.95	☐
Starquest + Encounter	£7.95	☐
Pixel Power	£7.95	☐
Harvester + Brainstorm	£7.95	☐

PLEASE SEND ME THE GAMES
AS TICKED
Total cheque/P.O. enclosed ________
Cheque payable to Quicksilva Limited.

NAME ....

ADDRESS ....

Please send your order to:
QUICKSILVA LIMITED,
PALMERSTON PARK HOUSE,
13 PALMERSTON ROAD,
SOUTHAMPTON.

WARNING: These programs are sold according to QUICKSILVA Ltd's terms of trade and conditions of sale. Copies of which are

Volume 2
No 1
Summer 1983

Editor: Henry Budgett
Assistant Editor:
Wendy J Palmer
Advertisement Manager:
Miriam Roberts
Advertisement Copy Control: Sonia Hunt
Managing Editor:
Ron Harris BSc
Managing Director: T J Connell

Origination and design by MM Design & Print

Personal Software is a quarterly magazine appearing on the third Friday in May, August, November and February.

Distribution by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS Tel: 01-247 8233. Printing by: Alabaster Passmore & Sons Ltd, Tovil, Maidstone, Kent.

The contents of this publication including all articles, designs, plans, drawings and programs and all copyright and other intellectual property rights therein belong to Argus Specialist Publications Limited. All rights conferred by the Law of Copyright and other intellectual property rights and by virtue of international copyright conventions are specifically reserved to Argus Specialist Publications Limited and any reproduction requires the prior written consent of the Company. © 1983 Argus Specialist Publications Limited.

Subscription Rates: UK £7.80 for four issues including postage. Airmail and other rates upon application to Personal Software Subscriptions, 513 London Road, Thornton Heath, Surrey CR4 6AR.

CONTENTS

Editorial & Advertisement Office
145 Charing Cross Road, London WC2H 0EE
Telephone: 01-437 1002 Telex: 8811896

Preface 4

Getting Converted 6
Hints on how to convert programs from one Commodore machine to another.

Leapfrog 13
Our micro version of that old pub game played with coins.

Power Boat 16
Have all the fun of power boating without feeling seasick!

The Valley 19
Save the kingdom of the Valley by combatting dragons, Balrogs and Wraiths. Choose your character with magic or physical strengths and do battle in our epic game, The Valley.

Towers of Brahma 38
Moving rings from one pillar to another may sound easy but just try it!

Micro Examination 40
Test your friends and children with this multiple choice program.

Quiz Time 45
Assess your performance in terms of speed and accuracy with a multiple choice program.

Multipurpose Records 50
Set up your own filing system which enables you to store, search, edit and retrieve data.

VIC Editor 54
Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.

Commodore Communications58
Get Commodores talking to each other using this application.

Address Book 63
Compile an address book, or any other similar list, and throw away those bits of paper.

Multicolumn Records 66
A multipurpose data base program for use at home or in the office.

Subroutine Library 74
A library of BASIC subroutines.

Toolkit Program 77
A simple toolkit program for the Commodore 64.

Tailoring VIC's Characters...79
Create your own characters on your VIC-20.

Maxi-Mander 84
Bomb-proofing your software against unskilled fingers.

VIC Blow Up 92
Find out how characters are made up and generate giant versions on the VIC-20.

Program Protection 95
Simple tips on how to protect your program from being easily copied.

Bibliography 97
A brief perusal of some of the multitude of books on Commodore computers.

Credits: Our grateful thanks to the following individuals for their assistance in the production of this issue. Enterprise Pictures Limited, Kieran, Adrian, Junior, Tom, Enzo, Wesley, Chris, Walter, John and Bill.

PREFACE

Welcome to this issue of *Personal Software* which is dedicated to the Commodore series of microcomputers. The series of machines includes the very popular VIC-20, the fairly new Commodore 64 and the 2000, 300 and 4000 range so we have tried to put together a variety of software that should appeal to Commodore users regardless of their particular machine. We say regardless of machine because we know how frustrating it can be when you see an interesting program for a VIC-20, say, and you sit gazing longingly at your Commodore 64 wishing you knew how to adapt the program for your micro. Here we have endeavoured to be of help: on page 6 of this magazine the Editor has put together some hints and techniques to show you how to convert the programs from one machine to another. So with only a little alteration you should be able to implement almost all of the programs in this issue on your Commodore microcomputer.

Armed with the knowledge that you really can use most of the programs in this magazine you will probably want to know exactly what there is, and we have tried to pull together some of the best material available for Commodore machines that have been published in ***Computing Today*** and have added some new specially commissioned features and programs to produce a varied package.

The material included in this issue ranges from the fairly elementary level to the more advanced stuff. A lot of VIC-20 users will probably be in the younger age groups and will no doubt head for the games programs first. Here they will find such games as Leapfrog and Towers of Brahma to test their abilities. To test them even further (and also very much for adult users) is our famed game The Valley. This was originally published in ***Computing Today*** and has developed an enthusiastic following, but it is a very large program and so you shouldn't be put off by its size. And if you really can't face keying in the whole thing you can buy the game ready for loading into your micro directly from ASP Software.

Data base programs are becoming something of an 'in-thing' at the moment so we have included such features as Multipurpose Records, Address Book and Multicolumn Records to help you set up your own data base, invaluable for all sorts of information storage and retrieval.

For the more educationally minded, Quiz Time and Micro Examination are included to help you set up your own system for testing knowledge, be it for fun or for more serious purposes (although I hear my teacher saying that education was fun!).

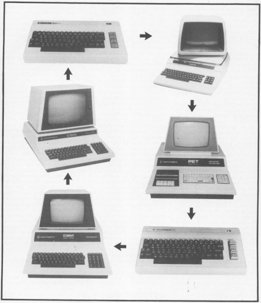
Those of you who like to delve deeper into their micros, rather than just keying in a program and accepting what it does, will be more interested in the utilities and applications programs herein. You can enlarge the VIC-20's characters using VIC Blowup, or you can alter the screen size by implementing our VIC Editor program. And for those of you with more than one Commodore machine at your disposal you can actually get them to pass programs from and to each other after reading Commodore Communications. Of course we should never forget the ever present problem of software security and we show a simple but effective way of protecting your own programs in our Program Protection feature.

Considering that there must be getting on for 250,000 Commodore micros out there in 'Micro Land' the number of books written on the machines and their software are more than a few. We have not attempted to present a comprehensive list of the books that are available since that would probably take a whole issue in itself, but we have taken a brief look at the books that we had to hand and that should at least give you a taste of what's around.

We know from bitter experience (and your enquiries) that there is nothing worse than spending hours typing in a program only to find that it doesn't work. So let us assure you here that we have checked these listings as carefully as possible before putting them into the magazine. And those that have previously appeared in ***Computing Today*** have been updated with any changes that were brought to our attention. Thus when you find your program doesn't work you should at least initially assume that it is correct in the magazine and you should check it **very** carefully before you contact us. It is often helpful to have someone check it who did not type it in since it is often difficult to see your own mistakes. If after checking the listing very carefully you still cannot find a typing error (and you have ensured that you are using the correct machine!) then it is better if you write to us rather than 'phoning in with your query since it gives us more time to try and help and also gives us more time to try to get the next issue of ***Personal Software*** together!

CONVERSIONS

Getting Converted — Hints on how to convert programs from one Commodore machine to another.



Henry Budgett

GETTING CONVERTED

Here we attempt to show you how to go about converting programs for one Commodore machine so that it will run on your particular one.

Commodore must be one of the very few microcomputer manufacturers who have tried to maintain at least some degree of compatibility between the various computers that they make. Before you can make any kind of start at adapting a program from this magazine for, let's say, a VIC-20 to run on a standard 3000 series PET there are a number of important points to be observed. As usual with a range of machines that has developed across a long period of time minor, subtle variations occur within each particular model — in this case we are concerned with the changes to the BASIC language ROMs. You can establish the particular variety of BASIC fitted to your machine just by looking at the screen of the machine when you turn it on.

TRIED AND TESTED

If you've got an original PET 2001-8 system, those were the ones with a calculator style keyboard, you've got BASIC 1.0 and, unless you've had the new ROMs fitted, certain POKEs to the operating system memory (addresses less than 1024) will definitely need to be changed. As we could fill the entire magazine with the list of these I suggest that you consult one of the books referenced at the end of this article! If your PET has a 'decent' keyboard or is labelled as being a CBM 3008, 3016 or 3032 then you've got another version of BASIC called BASIC 2.0. This is also the version of the language supplied on the VIC-20 and Commodore 64 systems. Owners of 4008, 4016, 4032 or 8000 series machines have yet another variant known as BASIC 4.0 which is equipped with built-in disc operating commands.

As if these changes weren't sufficient to make you doubt my earlier statement that Commodore machines were generally compatible with one another there is another, more substantial difference between certain models. This area of change is the size of the display screen and, as a direct consequence, the locations in memory left for the display to occupy. Original 2001-8 machines, the 3000 series, the 4000 series and the 'Fat Forty' 4000 models with a 12" display instead of the usual 9" one have a display format of 25 lines of 40 characters. The new Commodore 64 system also has a 25 by 40 screen format. When Commodore introduced the VIC-20 they decided, for reasons unknown, to provide a screen format of 23 lines of 22 characters. Not satisfied with this unusual format they also provided a second area of memory arranged in the same way to act as a colour memory. Table 1 shows the necessary details for all the current Commodore machines.

Having astounded programmers by coming up with the VIC-20 screen format the company attempted to redress the balance somewhat by giving the business oriented 8000 series machines a decent 25 lines of 80 characters. However, the launch of the Commodore 64 saw a return to the tried and tested 25 by 40 format. The upshot of this is that programs which require direct access to the screen memory by means of the PEEK and POKE commands will need substantial changes in this area. Once again Table 1 will be of assistance as this gives the various memory address for the screen and colour maps.

GETTING IT IN

Actually keying the programs contained within these pages into your Commodre machine is a simple task, if a little laborious. You can generally make any necessary changes as you go although a screen planning grid will be an invaluable aid for re-formatting the display. If you're loading a VIC-20 program into anything else other than the Commodore 64 you must omit all references to the second display memory otherwise wierd and wonderful things will happen as your program gaily POKEs numbers into its own memory! The VIC chip itself should also be marked down as a candidate for removal, once again the presence of Table 1 should prove helpful.

Programs which use specific facilities provided by the Operating System are generally well commented and in several cases suggestions will be made regarding changes between the various machines. Very specific commands to do with the input or output of information to cassette or disc are included in some of the programs, notably the data base systems and the two-pass assembler. The type of disc unit required is either specified in the text or the code is written in such a way that any of the various Commodore discs will work. If you don't have discs and want to make use of cassette storage instead simply look at one of the programs that uses cassette storage techniques and substitute these for the disc code. All the Commodore machines handle the cassette in the same way.

For those converting programs onto the VIC-20 a special word of caution. The amount of memory, and its location within the memory map, for the storage of programs can be a tricky point. Because there are a number of different sized RAM packs available: 3K, 8K, 16K etc you do need to know which variant the program has been written for! The problem is not that the program might not fit into the available RAM, although that obviously does come into it, but that the screen memory map moves around the RAM depending on how much extra memory you have plugged in. This can cause severe problems with PEEKs and POKEs to screen locations so you should be aware that if you are keying the program into a VIC-20 which has a different amount of RAM to that specified in the article you may well have to make changes.

LOADING FROM TAPE

The tape format used by Commodore is consistent right across the range of machines so if you have a tape of a VIC-20 program there should be nothing to stop you LOADing it into, say, a 3000 series CBM machine. The only problem is that both the VIC-20 and the Commodore 64 store their programs in a different place in memory to the various 'PET-type' machines. However, all is not lost as you can make use of the built-in monitor program on the PETs to resolve the problem. If you are working with a VIC-20 that has an extra 3K of RAM attached your programs will load correctly as though they were created on a PET, you'll still have to make the other changes though!

Programs that LOAD in but aren't there when you type LIST haven't been lost for good, they are merely hiding somewhere in memory. There are a number of ways to get the programs back to the right place but the following method is probably the neatest of all.

1) Ensure that the PET's memory is clear before you LOAD — turning it off is often the easiest method.
2) LOAD the VIC-20 or Commodore 64 program from tape.
3) Try to LIST it — you never know!
4) Assuming that you get no joy from LIST perform the following sequence of inputs, everything you need to type is **underlined**. The <CR> symbol simply means that you should press Carriage Return at this point.

```
READY
 SYS 1024 <CR>

b*    pc   irq   sr ac xr yr sp
.;  0401 e455 32 04 5e 00 f6
.
```

If the figures given under the letters pc, irq etc don't match yours it doesn't matter, they depend on what's been going on inside the machine and will not affect what we are about to do. Now, type in the following:

```
. m 0400 0400 <CR>
.:  0400 00 00 00 aa aa aa aa aa
.
```

Using the cursor control keys move the cursor up the screen so it rests over the first 0 after 0400 and then edit the line so that it looks like this:

```
.:  0400 00 01 08 01 00 0f 00 aa
```

Press <CR> and the cursor will move down to the next line. This example will link a brand new line of program at the bottom of memory to a Commodore 64 program which starts at 0800 Hex. The link byte is the second and third pair of digits after the 0400. If we wanted to link to a VIC-20 program that was generated on an unexpanded machine the line would have read:

```
.:  0400 00 01 10 01 00 0f 00 aa
```

Similarly, for an expanded VIC-20 whose programs start at 1200 Hex we would have edited the line to read:

```
.:  0400 00 01 12 01 00 0f 00 aa
```

Now, all that remains is to go back to BASIC so type the following:

```
. x <CR>
READY
```

5) Now, LIST the program. Surprised? Well, you created a new line of program within the computer's memory which you linked to the beginning of the program that you loaded from tape, that's why there's a new line 1.
6) As you don't want the new line 1 delete it by typing 1<CR> and then LIST the program again. As the computer had to delete a line of a program from its memory it needed to tidy up all the others and has kindly moved them all down into the correct part of memory for you. You can now modify, SAVE or RUN the program quite normally.
7) The purists among you will realise that we aren't quite finished as we should really alter the variable pointers. To do this is not that difficult, the following sequence shows how.

```
READY
 SYS 1024 <CR>

b*    pc   irq   sr ac xr yr sp
.;  0401 e455 32 04 5e 00 f6
. m 0400 5000 <CR>
```

The system will now proceed to dump the contents of its memory onto the screen at high speed! Use the 'slow down' key to reduce the speed to a manageable rate and watch for the apparently random selection of numbers to change to nothing but aa aa aa aa aa etc. At this point we have reached the end of the program. Stop the listing by pressing the RUN/STOP key and you should have something like this (once again the actual numbers don't matter):

```
.:  14a8 4f 43 41 54 45 44 20 44
.:  14b0 41 54 41 20 45 52 52 4f
.:  14b8 52 22 3a 89 32 30 31 30
.:  14c0 00 00 00 aa aa aa aa aa
.:  14c8 aa aa aa aa aa aa aa aa
.
```

Now enter the following:

```
. m 0028 0028 <CR>
```

The system will display something like:

```
.:  0028 01 04 c3 17 c3 17 c3 17
.
```

Using the cursor keys again edit the line to read:

```
.:  0028 01 04 c3 14 c3 14 c3 14
```

and press <CR>. You have ▶

now told the interpreter that the program ends at the location 14c3, the first occurrence of aa after the three pairs of 00 which indicate the end of a program. You have also informed the interpreter that it can store its variables in memory immediately after the end of the BASIC program. A quick tip for those with corrupted programs is to restore these three pairs of locations to the address the first byte after the end of the program, you lose your variables but you get your program back! All that remains is to return to BASIC by typing x <CR> and listing the program. The astute will have realised that I've made life much easier for the person attempting this for the first time by filling the computer's memory with nothing, indicated by the appearance of aa all through the memory. This makes it significantly easier to search for the end of a program, under normal circumstances you would have to scan for the 00 00 00 end of program marker which is a chore.

A MATTER OF STANDARDS

If you are unfamiliar with the system of standard symbols developed in *Computing Today* over the last four years to simplify the printing of graphics and cursor control characters then you may well be somewhat confused by the appearance of square brackets with such expressions as CLS inside them. We use this convention to overcome the problems of reproducing the special symbols that a number of computers generate to indicate such as the PET's reversed Heart character which indicates a Clear Screen command. Figure 1 gives the complete list of cursor control codes that you will find in the programs.

The Commodore range also sports a built-in set of pre-programmed characters which can be accessed directly from the keyboard or POKEd into place. Because these characters cannot be successfully reproduced we use a similar convention to that outlined above to include these in the listings. Figure 2 shows how it is arranged for the Commodore machines but take note that on the VIC-20 you have two graphics characters per key so the [G<] and [G>] conventions must be used here. The VIC-20 and Commodore 64 also allow colour commands to be embedded in PRINT statements. As these are generated by a control code sequence we show these as [CTL BLU] which indicates that at this point both the Control key and the Blue colour key should be pressed together. Both the VIC-20 and the Commodore 64 have four programmable function keys on the right of the keyboard and thse are represented by the [fnx] convention where x is a number betweeen 1 and 8.

[CLS]	CLear Screen
[HOM]	HOMe cursor
[CL]	Cursor Left
[CR]	Cursor Right
[CU]	Cursor Up
[CD]	Cursor Down
[REV]	REVerse video on
[OFF]	Turn it OFF
[SPC]	SPaCe
[CTL]	ConTroL key
[fn]	Function key
[G<]	Graphic left
[G>]	Graphic right

Fig. 1. The complete set of cursor control codes and their program representations.

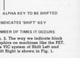


Fig. 2. The way we indicate block graphics on machines like the PET. The VIC system of Shift Left and Shift Right is shown in Fig. 1.

PET 2000/CBM 3000/CBM 4000	
Screen memory	32768 to 33767
Colour memory	Not applicable
BASIC starts at	1024 (0400 Hex)
CBM 8000	
Screen memory	32768 to 34767
Colour memory	Not applicable
BASIC starts at	1024 (0400 Hex)
Unexpanded VIC-20	
Screen memory	7680 to 8185
Colour memory	38400 to 38905
BASIC starts at	4096 (1000 Hex)
VIC-20 plus 3K	
Screen memory	7680 to 8185
Colour memory	38400 to 38905
BASIC starts at	1024 (0400 Hex)
VIC-20 plus 8K or 16K	
Screen memory	4096 to 4607
Colour memory	37888 to 38399
BASIC starts at	4608 (1200 Hex)
Commodore 64	
Screen memory	1024 to 2023
Colour memory	55296 to 56295
BASIC starts at	2048 (0800 Hex)

Table 1. The locations of the screen memory maps, colour memory maps and the start of BASIC programs for the various Commodore systems.

Bibliography
The PET Revealed by Nick Hampshire
PET Personal Computer Guide by Adam Osborne, Jim Strasma and Ellen Strasma
PET Graphics by Nick Hampshire

IT'S HERE!
Home Computing WEEKLY
ONLY 35p!

Buying or selling? Now's the time to take a look.

- **LATEST** computers featured and their performance assessed by Home Computing Weekly's team of resident experts.
- Tested program listings for all the populars including BBC, ZX80, ZX81, Spectrum, Atari, VIC-20, Oric and others.
- The **LATEST** news and views on personal computers.
- Inside information for the computer enthusiast.
- News coverage like you've never seen before.
- Over five pages of software reviews each week.

OUT NOW AND EVERY TUESDAY

Don't be left without — cut along dotted line and give this coupon to your newsagent.

Dear Newsagent,
Every Tuesday, please reserve me a copy of

Home Computing WEEKLY

Name: . . . .

Address: . . . .

. . . .

. . . .

Thank You

Let Commodore expand your horizons.

VIC 20 is the finest home computer that money can buy.

And the better you get to know it, the more confident, adventurous and ambitious you'll become.

You'll want to take advantage of the vast range of VIC software: a superb and constantly-growing selection of programs, embracing business systems, entertainment, education and many applications in the home.

Every program in the series has been designed by experts, and chosen for its quality and value for money.

VIC business software covers a wide range of applications, including spread-sheet analysis, stock control, information handling and word-processing.

A mind-blowing range of games including Scott Adams' world-famous 'Adventure' series.

Advanced space games, including the sophisticated 'Omega Race'.

Learn subjects as diverse as English Language, programming, and biology.

And 'home' software ranges from IQ tests to Robert Carrier menus.

In addition, there is a range of VIC software, like programmers' aids and graphics packages–

VIC 20
CARTRIDGE
SARGON II CHESS
commodore COMPUTER

to add to your understanding and enjoyment of computers and computing.

There's even a special 'VicSoft' Club for VIC 20 enthusiasts, with many advantages including special offers to club members.

VIC-20
SOFTWARE
CARTRIDGE
MATHEMATICS 2
KNOW YOUR PERSONALITY
MONEY MANAGER
MASTERMIND
Commodore COMPUTER

VIC software will expand your horizons. And your mind.

PRICES RANGE FROM £4.99 to £24.95 INC. VAT.

commodore
VIC 20

For more information, a catalogue of VIC software
and details of your local retailers or dealers please phone or
complete the coupon and send to:
The Commodore Information Centre,
675 Ajax Avenue, Slough, Berkshire SL1 4BG.
Telephone: Slough (0753) 79292.

Name
(MR./MRS./MISS)
Address

Postcode

VC PSO 06 83

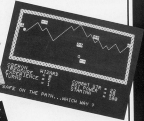
GAMES

Leapfrog — Our micro version of that old pub game played with coins.
Originally published in **Computing Today**, January 1982.

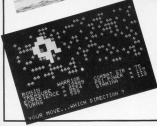
Power Boat — Have all the fun of power boating without feeling seasick.
Originally published in **Computing Today**, March 1981.

The Valley — Save the kingdom of the Valley by combatting Dragons, Balrogs and Wraiths. Choose your character with magic or physical strengths and do battle in our epic game, The Valley.
Originally published in **Computing Today**, April 1982.

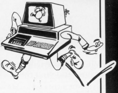
Towers of Brahma — Moving rings from one pillar to another may sound easy but just try it.
Originally published in **Computing Today**, August 1980.



```
OBERON        WIZARD
TREASURE    = 0
EXPERIENCE  = 5
TURNS       = 1
                        COMBAT STR = 22
                        PSI POWER  = 28
                        STAMINA    = 100
SAFE ON THE PATH...WHICH WAY ?
```



```
RONIN         WARRIOR      COMBAT STR = 75
TREASURE    = 12989        PSI POWER  = 117
EXPERIENCE  = 8654         STAMINA    = 113
TURNS       = 539
YOUR MOVE...WHICH DIRECTION ?
```



LEAPFROG

A game, its concepts, planning and implementation for the PET.

The program Leapfrog is about a relatively simple but very interesting board game which is often played in pubs with coins. In its usual form there are six coins and seven spots, as shown in Fig. 1.

The purpose of the game is to interchange the two sets of coins, and the rules are very simple:

1. The LEFT coins can move only to the RIGHT and the RIGHT coins can move only to the LEFT.
2. There are two possible types of move:
   (a) to an adjoining empty spot;
   (b) over another coin to an empty spot (like in checkers).

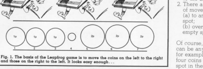


Fig. 1. The basis of the Leapfrog game is to move the coins on the left to the right and those on the right to the left. It looks easy enough . . .

VARIABLES

A$,B$P1,Z$	Temporary variables
SP1,SP(N)	Screen positions
C,C1,C2	Counters
A(N)	Record of where spot is, ie Left, Right or Middle
B(N)	Record of end status of points for checking
S	Character number for screen display
P1	Temporary variable for SP(N)
Z	Number of spots in game
ZL,ZM,ZR	Left, Middle and Right specific values of Z
CS,NS	Current spot and New spot. That is, numbers of spot's last and new positions
R$	Record of moves made.

PROGRAM STRUCTURE

100-160	Introduction
170-930	Sets up example on screen, describes game and its rules
940-1280	Chooses game and sets up appropriate screen display
1290-1450	Spot to be moved and where to, and checks legality of move
1460-1560	Moves Left to Right
1570-1670	Moves Right to Left
1680-1730	Spot is empty message
1740-1790	Spots too far apart message
1800-1850	Spot is not empty message
1860-1910	Wrong direction message
1920-2140	Checks for solution
2150	Makes spots on screen
2220	Numbers spots on screen
2250	Removes a large message from top of screen
2300	Removes a small message from top of screen
2350	Removes a message from bottom of screen
2400	Holds screen until key is pressed.

Of course, the number of spots can be any odd number so that, for example, there might be four coins on each side and a spot in the middle making nine spots altogether. The program allows you to choose between three, five, seven and nine spots.

GAME STRATEGY

There are two distinct advantages to playing the game on a computer VDU. In the first place, the initial stages of becoming clear about just what is permitted in a game are much more easily handled on a computer. Because, while it will obviously allow you to make wrong or foolish moves, it will not allow you to make illegal or incorrect moves. This means that the machine forces you to learn the rule structure of the game very quickly and forces you further into concentrating on strategy.

The second advantage is that it is possible for the computer to keep a record of all the moves made in any particular game. At the end of the game this record can be assessed and studied. In this way wrong moves can be detected, winning patterns become clear and the structure of the game emerges.

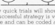
Like all games the initial motivation for playing is pleasure, and this should not be minimised in any way. So, at this stage all readers ought to go off, type in the game and ▶

play it for a while. But eventually a pattern of movements will begin to emerge and curiosity will develop about how the game might be symbolised and its structure analysed. When that stage is reached the educational and, in this case mathematical, aspect of the game begin to become more interesting and that is the ideal outcome. That is to say, not only is there pleasure to be gained, but eventually there is an educational payoff.

PROGRAM ANALYSIS

This next section is an attempt to analyse and generalise this particular game and the program can be used at each stage to test the results and conclusions. But remember that this is a second-level activity and ought to follow some experience at playing the game. When you read this next part you will know how to succeed at the game and how to generalise and the interest will be purely academic.

The simplest version of the game looks like this:

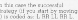


A few quick trials will show that the successful strategy is very simple and can be coded as: LRL (RLR).

This means:

First move the L from 1 to 2.
Then move the R from 3 to 1.
Then move the L from 2 to 3.

The next version of the game looks like this:



In this case the successful strategy (if you start by moving 2) is coded as: L RR LL RR L. This means:

First move an L piece.
Then move two R pieces.
Then move two L pieces.
Then move two R pieces.
Finally move an L piece.

```
100 A$="[REV]---LEAPFROG---[OFF]":R$=""
110 PRINT CHR$(147);TAB((40-LEN(A$))/2);A$
120 INPUT "[CD][5 CR][REV]DO YOU WANT INSTRUCTIONS.[OFF][2 CR]*[3 CL]";B$
130 IF LEFT$(B$,1)="Y" THEN 170
140 IF LEFT$(B$,1)="N" THEN 940
150 PRINT "TRY AGAIN"
160 GOTO 110
170 PRINT "[CD]YOU CAN HAVE 3, 5, 7 OR 9 POINTS IN"
180 PRINT "[CD]EACH GAME. WE WILL DEMONSTRATE WITH 7"
190 PRINT "[CD]AND THEN YOU CAN CHOOSE."
200 GOSUB 2410
210 PRINT CHR$(147)
220 SP1=33052:REM ** SCREEN POSITION 1
230 REM ** SET UP SEVEN POINTS
240 FOR C=1 TO 7
250 SP(C)=SP1+4*(C-1)
260 NEXT C
270 FOR C=1 TO 3:REM ** LEFT HAND POINTS
280 A(C)=1:B(C)=2
290 NEXT C
300 A(4)=0:B(4)=0:REM ** MIDDLE POINT
310 FOR C=5 TO 7:REM ** RIGHT HAND POINTS
320 A(C)=2:B(C)=1
330 NEXT C
340 GOSUB 2310
350 PRINT "[HOM]FIRST OF ALL THERE ARE 7 SPOTS."
360 PRINT "[CD][4 SPC][REV]LIKE THIS:-[OFF]"
370 S=46
380 FOR C=1 TO 7
390 P1=SP(C)
400 GOSUB 2160
410 GOSUB 2230
420 NEXT C
430 GOSUB 2410
440 GOSUB 2360
450 GOSUB 2310
460 PRINT "[HOM]THERE ARE 3 [REV]L[OFF] BLOCKS ON"
470 PRINT "[5 CR]THE LEFT. [REV]LIKE THIS:-[OFF]"
480 S=12
490 FOR C=1 TO 3
500 P1=SP(C)
510 GOSUB 2160
520 NEXT C
530 GOSUB 2410
540 GOSUB 2360
550 GOSUB 2310
560 PRINT "[HOM]THERE ARE 3 [REV]R[OFF] BLOCKS ON"
570 PRINT "THE RIGHT....[REV]LIKE THIS:-[OFF]"
580 S=146
590 FOR C=5 TO 7
600 P1=SP(C)
610 GOSUB 2160
620 NEXT C
630 GOSUB 2410
640 GOSUB 2360
650 GOSUB 2310
660 PRINT "[HOM]YOUR TASK IS TO MOVE THE [REV]L[OFF] BLOCKS TO"
670 PRINT "WHERE THE [REV]R[OFF] BLOCKS ARE AND VICE VERSA"
680 PRINT "SO THAT THEY END UP [REV]LIKE THIS:-[OFF]"
690 GOSUB 2410
700 GOSUB 2360
710 S=146
720 FOR C=1 TO 3
730 P1=SP(C)
740 GOSUB 2160
750 NEXT C
760 S=12
770 FOR C=5 TO 7
780 P1=SP(C)
790 GOSUB 2160
800 NEXT C
810 GOSUB 2410
820 GOSUB 2360
830 PRINT CHR$(147)
840 PRINT "THERE ARE THREE RULES:-"
850 PRINT "[CD]1. LEFT BLOCKS CAN MOVE ONLY"
860 PRINT "[3 CR]TO THE RIGHT AND RIGHT BLOCKS"
870 PRINT "[3 CR]CAN ONLY MOVE TO THE LEFT"
880 PRINT "[CD]2. A BLOCK CAN BE MOVED TO AN"
890 PRINT "[3 CR]ADJOINING SPOT IF THIS IS EMPTY"
900 PRINT "[CD]3. A BLOCK CAN JUMP OVER AN"
910 PRINT "[3 CR]ADJOINING BLOCK AS IN CHECKERS"
920 GOSUB 2410
930 GOSUB 2360
940 PRINT CHR$(147)
950 INPUT "HOW MANY SPOTS DO YOU WANT (3,5,7,9)[2 CR]*[3 CL]";Z
960 IF Z<3 OR Z>9 THEN 940
970 IF Z=3 OR Z=5 OR Z=7 OR Z=9 THEN 990
980 GOTO 940
990 SP1=33048:REM ** SCREEN POSITION 1
1000 REM ** SET UP Z POINTS
1010 ZL=(Z-1)/2:ZM=(Z+1)/2:ZR=(Z+3)/2
1020 FOR C=1 TO Z
1030 SP(C)=SP1+4*C-2*Z+17
1040 NEXT C
1050 FOR C=1 TO ZL:REM ** LEFT HAND POINTS
1060 A(C)=1:B(C)=2
1070 NEXT C
1080 A(ZM)=0:B(ZM)=0:REM ** MIDDLE POINTS
1090 FOR C=ZR TO Z:REM ** RIGHT HAND POINTS
1100 A(C)=2:B(C)=1
1110 NEXT C
1120 PRINT CHR$(147)
1130 S=12
1140 FOR C=1 TO ZL
1150 P1=SP(C)
1160 GOSUB 2160
1170 GOSUB 2230
1180 NEXT C
1190 S=46
1200 P1=SP(ZM)
1210 GOSUB 2160
1220 GOSUB 2230
1230 S=146
1240 FOR C=ZR TO Z
1250 P1=SP(C)
1260 GOSUB 2160
1270 GOSUB 2230
```

```
1280 NEXT C
1290 GOSUB 2310
1300 PRINT"[HOM]IF AT ANY TIME YOU WISH TO STOP PRESS 99"
1310 GOSUB 2410
1320 GOSUB 2360
1330 GOSUB 2310
1340 PRINT"[HOM]CHOOSE THE NUMBER OF THE [REV]BLOCK[OFF] THAT YOU"
1350 INPUT "WISH TO MOVE[2 CR]*[3 CL]";CS:REM ** CURRENT SPOT
1360 IF CS=99 THEN 2100:REM ** END
1370 IF A(CS)=0 THEN 1680:REM ** SPOT IS EMPTY
1380 GOSUB 2310
1390 PRINT"[HOM]CHOOSE THE NUMBER OF THE [REV]SPOT[OFF] TO WHICH YOU"
1400 INPUT "WISH TO MOVE[2 CR]*[3 CL]";NS:REM ** NEW SPOT
1410 IF NS=99 THEN 2100:REM ** END
1420 IF ABS(CS-NS)>2 THEN 1740:REM ** MORE THAN TWO APART
1430 IF A(NS)<>0 THEN 1800:REM ** SPOT NOT EMPTY
1440 IF A(CS)=2 THEN 1570:REM ** RIGHT TO LEFT
1450 IF A(CS)=1 THEN 1460:REM ** LEFT TO RIGHT
1460 REM ** MOVE LEFT TO RIGHT
1470 IF NS<=CS THEN 1860:REM ** WRONG WAY
1480 R$=R$+"L"
1490 S=46:P1=SP(CS)
1500 GOSUB 2160
1510 S=12:P1=SP(NS)
1520 GOSUB 2160
1530 A(CS)=0:A(NS)=1
1540 PRINT"[REV]NOW NEXT MOVE[OFF]"
1550 GOSUB 2310
1560 GOTO 1930:REM ** CHECKS IF COMPLETE
1570 REM ** MOVE RIGHT TO LEFT
1580 IF NS>=CS THEN 1860:REM ** WRONG WAY
1590 R$=R$+"R"
1600 S=46:P1=SP(CS)
1610 GOSUB 2160
1620 S=146:P1=SP(NS)
1630 GOSUB 2160
1640 A(CS)=0:A(NS)=2
1650 PRINT"[REV]NOW NEXT MOVE[OFF]"
1660 GOSUB 2310
1670 GOTO 1930:REM ** CHECKS IF COMPLETE
1680 REM ** SPOT EMPTY
1690 PRINT "[REV]THERE IS NO BLOCK ON THIS SPOT[OFF]"
1700 GOSUB 2410
1710 GOSUB 2360
1720 GOSUB 2260
1730 GOTO 1340
1740 REM ** MORE THAN 2 SPOTS APART
1750 PRINT"[REV]THE SPOTS ARE TOO FAR APART[OFF]"
1760 GOSUB 2410
1770 GOSUB 2360
1780 GOSUB 2260
1790 GOTO 1340
1800 REM ** SPOT NOT EMPTY
1810 PRINT"[REV]THIS SPOT IS NOT EMPTY, TRY AGAIN[OFF]"
1820 GOSUB 2410
1830 GOSUB 2360
1840 GOSUB 2260
1850 GOTO 1340
1860 REM ** WRONG DIRECTION
1870 PRINT"[REV]YOU CANNOT MOVE IN THAT DIRECTION, TRY AGAIN[OFF]"
1880 GOSUB 2410
1890 GOSUB 2360
1900 GOSUB 2260
1910 GOTO 1340
1920 REM ** CHECK IF COMPLETE
1930 GOSUB 2260
1940 FOR C=1 TO Z
1950 IF A(C)<>B(C) THEN 1970
1960 GOTO 1990
1970 C=7
1980 GOTO 1330:REM ** NOT COMPLETE
1990 NEXT C
2000 REM ** COMPLETED
2010 GOSUB 2310
2020 PRINT CHR$(147);"[14 CD]CONGRATULATIONS"
2030 PRINT"YOU HAVE SUCCEEDED"
2040 INPUT "[CD]DO YOU WISH TO SEE YOUR SOLUTION[2 CR]*[3 CL]";Z$
2050 IF LEFT$(Z$,1)="Y" THEN 2080
2060 IF LEFT$(Z$,1)="N" THEN 2140
2070 GOTO 2040
2080 PRINT"[15 CR][5 CD]";R$
2090 GOTO 2140
2100 INPUT "[17 CD]DO YOU WISH TO SEE YOUR SOLUTION[2 CR]*[3 CL]";Z$
2110 IF LEFT$(Z$,1)="Y" THEN 2130
2120 IF LEFT$(Z$,1)="N" THEN 2140
2130 PRINT"[15 CR][5 CD]";R$
2140 END
2150 REM ** MAKES SPOTS
2160 FOR C1=0 TO 200 STEP 40
2170 FOR C2=P1 TO P1+1
2180 POKE C1+C2,S
2190 NEXT C2
2200 NEXT C1
2210 RETURN
2220 REM ** NUMBERS POINTS
2230 POKE C1+C2+70,C+48
2240 RETURN
2250 REM ** CLEARS TOP OF SCREEN, LARGE
2260 FOR C=32760 TO 33007
2270 POKE C,32
2280 NEXT C
2290 RETURN
2300 REM ** CLEARS TOP OF SCREEN, SMALL
2310 FOR C=32768 TO 32927
2320 POKE C,32
2330 NEXT C
2340 RETURN
2350 REM ** CLEARS BOTTOM OF SCREEN
2360 FOR C=33680 TO 33708
2370 POKE C,32
2380 NEXT C
2390 RETURN
2400 REM ** HOLDS SCREEN
2410 PRINT "[HOM][23 CD][5 CR][REV]PRESS ANY KEY[OFF]"
2420 GET Z$
2430 IF Z$="" THEN 2420
2440 RETURN
```

Listing 1. The program for Leapfrog.

Notice that the number pattern is 122 2 1. The strategies for versions of the game with three, five and seven spots are as follows:

LRL
L RR LL RR L
L RR LLL RRR LLL RR L

Turn these into number patterns as before:

111
12221
1233321

and the nine spot version becomes, one would guess,

123444321

This means, begin by moving one L piece, then two R pieces, then three L pieces, and so on. You can try this out on the nine spot version on the computer and it does work.

At this stage the pattern is so simple and obvious that it is quite easy to succeed at any version of the game. But its analysis is not quite complete. The general case can be symbolised as the case with 2 N + 1 spots. That is, N spots on the left, N spots on the right, and one spot in the middle. So the pattern of movements can be shown as:

123 . . . NNN . . . 321

That is, one LEFT move, two RIGHT moves, three LEFT moves and this continues until N moves are necessary. This number of moves ie, N occurs three times, and then the number begins to reduce by one each time until one move only is necessary. At this stage the game is over.

The final question may well be, what is the total number of moves necessary at each level of the game, and this generalised pattern can be used to answer this. In the general case, the total number of moves is:

$$(1+2+3+\ldots+N)+N+$$
$$(N+\ldots+3+2=1)$$

That is, $2(N)+N=N(N+2)$.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Andrew Thomas & Gareth Callister

POWER BOAT

If you suffer from seasickness or you don't like getting your feet wet, this game should provide the excitement without the discomfort!

16
QEC Computers

OK 'HANDS UP' how many of you are dry-land sailors, or have trouble with navigational aids? If you are one, then here is the ideal program for you, it saves you from getting your feet wet — or *does it?*

You are in complete(?) control of an extremely powerful, fast power boat; in a race against other top power boat champions. The course and weather conditions vary on each game, though the weather conditions will not alter during the game. The course is marked out with bouys which will give you the direction and the maximum speed for that particular section (there are 20 sections in all). The direction is given in the form of the radius of the necessary turning circle (the program assumes that you are intelligent enough to know which way to turn).

To make the turn you must enter in the amount of RUDDER you need. Too much and you will overshoot the corner though 'slightly' cutting the corner will help you gain distance and position.

Both of the above will gain you immediate disqualification, just to hinder you. The lower the number entered for the rudder the smaller the turning circle and hence the smaller the radius. There is, however, one special case — '0' which will

```
100 PRINT CHR$(147);TAB(10);"POWER BOAT RACE"
110 PRINT TAB(10);"---------------"
120 PRINT
130 PRINT "YOU ARE IN CHARGE OF A FAST MOTOR BOAT"
140 PRINT "WITH A 20 SECTION COURSE TO COMPLETE."
150 PRINT
160 PRINT "YOU ONLY HAVE 2 CONTROLS ON THE BOAT:-"
170 PRINT
180 PRINT "1--THE POWER TO THE MAIN DRIVE (NOTE"
190 PRINT "  THIS IS NOT YOUR SPEED)."
200 PRINT "2--RUDDER CONTROL TO STEER YOUR BOAT"
210 PRINT "  (NOTE THE EFFECTIVENESS ALTERS WITH"
220 PRINT "  SPEED)."
230 PRINT
240 PRINT "[REV]WARNING[OFF] IF YOU OVER OR UNDER STEER YOUR"
250 PRINT "BOAT WILL CRASH."
260 PRINT
270 PRINT "EXCEEDING THE MAXIMUM SPEED BY TOO MUCH"
280 PRINT "WILL ALSO CRASH YOUR BOAT."
290 PRINT
300 PRINT "YOU ONLY HAVE A LIMITED AMOUNT OF FUEL"
310 PRINT "ON BOARD."
320 PRINT
330 PRINT "HIT ANY KEY TO CONTINUE"
340 GET A$
350 IF A$="" THEN 340
360 PRINT CHR$(147)
370 FOR I=1 TO 6
380 READ A$(I)
390 NEXT I
400 W=INT(RND(1)*6)+1
410 S=0
420 PRINT "CONDITION IS ";A$(W)
430 F=1000:REM ** FUEL
440 P=5:REM ** POSITION
450 D=10:REM ** DISTANCE TO LEADER
460 E=0
470 FOR X=1 TO 20
480 PRINT:PRINT:PRINT
490 IF RND(1)>0.7 THEN R=0:M=55+INT(RND(1)*55)-W*2:GOTO 520
500 R=INT(RND(1)*90)+10
510 M=INT(30+RND(1)*20+R/3)-W*2
520 PRINT "SECTION ";X
530 PRINT "POSITION ";P
540 IF D<0 THEN PRINT "YOU LEAD BY ";
    ABS(INT(D));" METRES":GOTO 560
550 PRINT "DISTANCE TO LEADER ";INT(D);" METRES"
560 PRINT:PRINT
570 PRINT "SPEED  MAX SPEED   RADIUS      FUEL"
580 PRINT INT(S),INT(M),INT(R),INT(F)
590 PRINT
600 INPUT "POWER TO MAIN DRIVE ";E1
610 IF E1>=100 OR E1<0 THEN PRINT "I SAID ";:GOTO 600
620 INPUT "RUDDER POSITION ";T
630 IF T>=10 OR T<0 THEN PRINT "I SAID ";:GOTO 620
```

Photography courtesy Enterprise Pictures Limited.

```
640 PRINT
650 IF E1-E>50-W*2 THEN 1000
660 C=(S/100+1)*10*T-R
670 IF ABS(C)>18-W THEN 1030
680 IF C<6 THEN 710
690 PRINT "YOU TOOK THE CORNER FAR TOO WIDE!!"
700 C=12
710 IF C>-6 THEN 740
720 PRINT "YOU CUT THAT CORNER TOO CLOSE!!"
730 C=18
740 E=E1
750 S=S/3+E/1.2-(T/10)
760 IF S<1 THEN S=1
770 IF S>1.1*M+10-W THEN 1100
780 D=D+M-S+C/2
790 P=INT(D/10+RND(2)+2)
800 IF P<2 THEN P=2
810 IF D<=0 THEN P=1
820 F=F-((E/10)↑2+E)/2
830 IF F<0 THEN 1130
840 NEXT X
850 IF P>1 THEN 890
860 PRINT "CONGRATULATIONS!! YOU NOT ONLY"
870 PRINT "FINISHED BUT ACTUALLY WON THE RACE."
880 GOTO 1150
890 IF D>30 THEN 930
900 PRINT "YOU FINISHED THE RACE AND WERE NOT FAR"
910 PRINT "BEHIND THE LEADER."
920 GOTO 63999
930 PRINT "WELL AT LEAST YOU FINISHED THE RACE."
940 PRINT:PRINT
950 PRINT"YOU FINISHED ";P
960 PRINT
970 PRINT "DISTANCE TO THE LEADER WAS ";INT(D);" METRES."
980 GOTO 1150
990 DATA CALM,MILD,CHOPPY,ROUGH,VERY ROUGH,STORMY
1000 PRINT "YOUR BOAT CAPSIZED FROM TOO MUCH"
1010 PRINT"ACCELERATION!!"
1020 GOTO 1150
1030 PRINT "CRASH!! YOU WENT THROUGH THROUGH THE MARKERS"
1040 PRINT "AND WERE IMMEDIATELY DISQUALIFIED!!"
1050 IF C>0 THEN 1080
1060 PRINT "YOU SHOULD NOT TAKE THE CORNER SO WIDE!"
1070 GOTO 1150
1080 PRINT "YOU SHOULD NOT CUT THE CORNERS!"
1090 GOTO 1150
1100 PRINT "CRACK!! YOU DAMAGED THE BOAT BY GOING"
1110 PRINT "SO FAST!!"
1120 GOTO 1150
1130 PRINT "PHUT!! I AM AFRAID YOU HAVE JUST RUN OUT"
1140 PRINT "OF FUEL!!"
1150 PRINT
1160 INPUT "DO YOU WANT ANOTHER GAME";Q$
1170 IF LEFT$(Q$,1)="Y" THEN PRINT CHR$(147):GOTO 400
1180 END
```

Listing 1. The program for the power boat race game.

cause your boat to go straight. This has suprisingly been done to make the navigation easier, but be warned! The ratio between the rudder and the radius varies from 1 to 10 when stationary, to 1 to 5 at top speed just to make the game a ***little*** more difficult. To complete the course you must also regulate your speed to a value near to or, preferably, slightly above the maximum speed for the section.

The power to the drive is ***not*** the speed, but the amount of power you supply to the drive. The more power you supply to the drive the faster the boat will go. If you put too much power in, you run the risk of converting your boat into an aeroplane and flipping over. Life jackets are compulsory whilst using this program, in case you get a bit too excited!

The program is written in a standard PET Microsoft BASIC and should present no real difficulty in converting to other machines. Note that PRINT CHR$(147) is a Clear Screen command and line 340 can be replaced with 340 INPUT A$ — line 350 should obviously be deleted.

For those of you who wish to alter the game the following list of variables may help.

C	is the difference between actual and rated turning circles.
D	is the distance to the leader.
E	is the previous 'power to the main drive'
E1	is the present 'power to the main drive'.
F	is the amount of fuel remaining.
M	is the maximum speed for the section.
P	is your position.
R	is the rated turning circle.
S	is your present speed.
T	is the rudder position.
W	is the prevailing weather condition.
X	is the section number.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Personal SOFTWARE

Personal Software is a quarterly publication dedicated to all aspects of software for the most popular micros.

Carrying on the trend of our system oriented issues the Autumn edition of **Personal Software** will contain a veritable host of material for the Apple II, Apple II+, and Apple IIe. So if you've got an Apple system you simply can't afford to miss this great issue.

Personal SOFTWARE
Summer 1983
Programs galore for your... VIC-20, PET, 8000 machines, and Commodore 64
Ingenious utilities to run
Educational software to test the family
Data-base for better business
Great games to play
BRITAIN'S BEST BUY FOR THE COMMODORE USER

Personal Software can be ordered directly from us at £7.80 per annum or £1.95 per copy. To ensure a single copy or a complete year's supply, fill in the form below, cut it out and send it with your cheque or postal order (made payable to ASP Ltd) to:

Personal Software
Subscriptions,
513 London Road,
Thornton Heath,
Surrey CR4 6AR.

– you can even spread the load with your credit card!

Don't miss out – subscribe now.

SUBSCRIPTION ORDER FORM

Cut out and SEND TO : Personal SOFTWARE

513 LONDON ROAD,
THORNTON HEATH,
SURREY,
CR4 6AR.

POST FREE SUBSCRIPTION

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss) ....
delete accordingly

Address ....

....

....

Signature ....

Date ....

Please commence my subscription to Personal Software with the .... issue.

SUBSCRIPTION RATES (tick ☐ as appropriate)

£7.80 for 4 issues U.K. ☐

£1.95 for a single copy of the issue ☐

I am enclosing my (delete as necessary) Cheque/Postal Order/International Money Order for £..........
(made payable to ASP Ltd)
OR
*Debit my Access/Barclaycard**
*(*delete as necessary)*

Insert card no. ☐☐☐☐☐☐☐☐☐☐☐☐☐

THE VALLEY

Join the thousands of brave adventurers who have dared to enter our multi-scenario role-playing game.

The news spread quickly throughout the lands of Tybollea. Vounim, mightiest wizard of the Northern Reaches, had arrived at the gates to the Princess Evanna's castle offering his aid against the Selric hordes which besieged her realm.

Her magical powers alone too weak to vanquish the foes, Princess Evanna eagerly accepted Vounim to her side. Together forming a psychic bond they wove a spell, powerfully constructed from the forces of light and darkness, to drive the savages from Tybollean soil. Their combined magicks scoured the Northern border-lands, scattering the enemy's host and laying waste the Selric threat forever.

In gratitude, Princess Evanna invited Vounim to make his home in her Kingdom and bestowed upon him the title 'Lord of the Valley Between Two Castles'. Knowing the land between the Princess' castle and her brother Xeron's to be most beautiful country, Vounim accepted the honour and began plans to build two strongholds in the forests of the Valley.

Time passed . . . far away from the village settlements, Vounim's Lairs (as his strongholds had become known) were often be the subjects of whispered conversation in the ale-houses of Tybollea. Even the Princess Evanna's councillors felt that the Princess had closed her eyes to the changes that had overtaken Vounim during the years he had attended the castle as her chief advisor. He had become quiet and withdrawn, only visiting the castle at the dead of night. It was even rumoured he had entertained in his strongholds members of the White Order, an evil brotherhood of wizards from the Southern Slopes.

Following just one of these visits from the White Order, Vounim had begun building two temples dedicated to the worship of an obscene lizard-like god, Y'Nagioth. Shrouded by evil swamps, it seemed as though none could stop the wizard from carrying out his ancient sacrificial rituals. At first, the Princess listened to stories of livestock dissappearing and of children running off with an air of humour; but soon even she could not deafen her ears to the allegations of the high taxes and cruelty of which her people complained. However, it was only when her war-like brother, Xeron, seemed to wither away in his sick-bed from the 'medicines' administered by Vounim that Princess Evanna began to see the threat posed to her throne.

Arranging a Council of War with her neighbouring Lords, Princess Evanna asked them to pledge their allegiance and grant her the aid she needed to crush the evil wizard. There was much brave talk and long discussion but eventually the Lords decided not to intervene. The worship of Y'Nagioth had spread and the peoples of Tybollea would likely as not support the wizard, High Priest of the lizard faith, rather than their over-lords.

The Lords quite clearly feared Vounim more than the Princess and rather than follow their heart's dictates chose the easy route. The Princess was disheartened and, clearing them from her Council Chamber, slumped into her throne deep in thought. She could destroy this wizard, she mused, but at what cost?

As dawn broke, the Princess' meditations were interrupted by a young wizard by the name of Alarian. She recognized him instantly as the novice attached to Baron Niitall, Lord of the Eastern Plains — one of the Lords she had expected would grant her the aid she would need. Although far below the Princess in magical prowess, Alarian was able to offer the wealth of his experiences as a youth apprenticed to the mighty Vounim back in the Northern Reaches. The young wizard also gave the Princess his copper amulet, studded with six precious gems — Alarian's amulet was a magical device, providing its wearer with the gift of life after mortal death.

Mounting her horse at the castle gates, Princess Evanna made one last desperate attempt to encourage her people to her side; the Lords looked away and her subjects jeered. So, muttering a curse, the Princess Evanna set off to face Vounim in his lair.

As she rode, she was saddened by the apparent sickness that hung over the Valley; nothing grew there now, save in the forests and swamps that surrounded Vounim's Lairs and the Temples of Y'Nagioth. Yet as she rode on she discovered, sheltered in the depths of the Valley floor, another building — a six-storey tower. She recognized the tower with sickening rapidity, she had once seen it in her youth — it was a replica of the Black Tower of Zaexon, the home of the brotherhood of the White Order. Satisfying herself that the tower was empty, she spurred her mount and raced with renewed vigour towards the demon wizard, Vounim.

Catching the wizard amidst a ghastly blood rite, Princess Evanna began casting a spell of banishment on Vounim. Caught off guard, the Lord of the Valley, screaming vile obscenities, started to fade from sight. With a final blood-curdling scream, he made a final gesture at the Princess before passing from the mortal plane. The Princess, surrounded by dancing lights, fell to the floor writhing in pain. She had been poisoned by Vounim's magick and, with mounting horror, realised this would be a magical and not a mortal death — the Amulet of Alarian would not help the Princess to cheat death.

Crumpled on the floor of Vounim's Lair, the Princess began to make her last magicks. She hid the Amulet in one of the Temples of Y'Nagioth and three of the stones she placed on the third floor of the Black Tower of Zaexon, the fourth stone on the fourth floor, the fifth and the sixth stones cached on the two top floors. Struggling to keep her consciousness, the Princess made one last gesture at the Helm and as she died, her magick passed into the Helm as it disappeared from sight forever.

The Valley buildings disappeared soon after Vounim's banishment, following him into the ethereal limbo in which the Princess had imprisoned him. Gradually over the years, the Valley returned to its former splendour. Alarian, satisfied that Evanna's spell was well cast, remained there for many years keeping his eye over the Kingdom. Then one fine morning, the first of Spring, Alarian, leaving a spell of watchfulness over the land, left for other adventures.

◆ ◇ ◆ ◇ ◆ ◇ ◆ ◇ ◆

Concluding the story, he turned his attention to the Valley lying far beneath his window, blanketed in swirling mists shrouding all but the highest tree tops. On the horizon, clearly silhouetted against the morning sun shone the silvery towers of Castle Xeron nestling on the hill many leagues away. All was still . . . almost peaceful.

"Listen, old man. I've heard your faerie story — just what is all this about?"

At the sound of the gruff voice, the hooded figure at the window swivelled around using his stick for support and, contemplating the six figures seated around his desk, began the slow and painful journey back to his chair.

"It is no faerie story, my friend" the old man muttered as he eased his back against the oaken carvings of the chair back. "I know the tale to be true for I was that young wizard, Alarian. It was I who, tens of thousands of years ago, sat with Princess Evanna helping her to prepare for her battle with Vounim".

Alarian lifted his hand weakly silencing the doubting questions of the company.

"Please listen. You would not understand the ways I have prolonged my life, so do not ask. Accept simply that I am Alarian and all I speak of is true. The spell of watchfulness I cast so long ago has called me here to protect your lands from great danger."

"With all due respect, Sire, are you not a little late?" said another of the figures, a novice wizard by his appearance. "The Valley has been a place of rumour and mystery, concealed by strange mists for nigh on thirty years".

"I'm afraid" sighed Alarian "that you will find out one day soon that not all magick works as effectively as you would wish. I believe my spell of watchfulness was weakened in much the same way as Princess Evanna's spell of banishment. During my time of apprenticeship to Vounim, I too formed a psychic link with my master hoping to amass power before my time. As the fates would have it, Vounim, through the past mystic bonds with the Princess and I, was able to divert much of the strength of our spells, allowing him to attempt a return to the mortal plane unseen.

"Vounim, Lord of the Valley, is smashing a pathway from the chaos of his world of banishment through to our own. In his present situation, halfway between chaos and reality, he is almost visible to me; his followers and his buildings, the Lairs, the Temples and the Black Tower, are already becoming reality again.

"As my wild-eyed barbarian friend pointed out earlier, I am but a frail old man. I can offer nothing but magical aid as I am all but restricted to this chair. Were I stronger, nothing would stop me fulfilling the quest alone but alas, it is to you I look for favour. Will any of you enter the Valley in search of the missing Helm of Evanna in my place?"

At the mention of the legendary Helm, the six figures moved closer around the wizard's desk.

"I can help whosoever decides to go" continued Alarian "but I can help only one of you at a time. I can create a path of safety between this castle and Castle Xeron, both of which will prove safe havens during your quest; I can also make the buildings visible to you —

although this means you will be seen and thus attacked by the inhuman creatures loyal to Vounim.

"You will need great experience to find the Helm of Evanna; such was Evanna's curse on her people — they spurned her when she needed their help to defeat Vounim. Princess Evanna hid the means to conquer any threat to the Kingdom so that only the bravest Tybollean could ever find it. To gain this experience, you would be wise to first search out my Amulet in one of the Temples of Y'Nagioth and, once found, journey to the Black Tower of Zaexon where you will find the six stones that fit the Amulet. However, care must be taken to find the stones in the correct order — if you don't, you will find they do not fit and will be useless to you.

"Although I have had little contact with my Amulet over past centuries, I am confident I can illuminate areas of residual magick within the Temples and the Black Tower indicating where magical items have been hidden at some time in the past. I will do my best to show you where the Amulet stones have been cached, but I have found that in my latter years I have not the concentration I used to have and you may find only worthless baubles instead — I will do my best";

"Well that's not good enough for me" cried a thief-like character jumping to his feet, "I'm damned if I'll follow you through this Valley — I've heard stories of the creatures who dwell there. Sorcery — hah!" He spat at Alarian's feet and departed.

As the slam the chamber door died away, Alarian surveyed the five remaining faces, a barbarian, a novice wizard, a cleric, a thinker and a warrior. Hand-picked and all native Tybolleans, Alarian wondered if one of these could achieve the impossible and bring back the lost Helm of Evanna.

"I would not blame you for following him," said Alarian, "the dangers he spoke of are all too real. Over the past weeks I myself have seen through my enchanted glass Dragons, Balrogs, Wraiths, even a creature with the very likeness of Y'Nagioth herself, a Thunder Lizard, roaming the Valley.

"You will not, however, enter the Valley unprotected. I will teach you a potent sleep spell and, as you gain experience, will be able to bestow two other spells on you: a mind lance to attack creatures with a high psychic power and a spell which attacks using the very Fires of Hell. However, you will use these spells sparingly as they are extremely dangerous in the hands of the untutored and it takes many years of study before a spell can be cast with no loss of stamina.

"You will doubtless have realised that I am no fighting man" continued the wizard, nodding respectfully towards the warrior and the barbarian, "but if I may offer some advice on hand-to-hand combat. There are three effective ways to fight a creature of great physical strength either strike its head, body or limbs. Obviously an attack to the limbs or body will eventually lead to success, but it may initially cause little damage. A strike to the head may kill the beast in one blow, but will leave you open to return blows while striking. The decision will be yours, I cannot help. However, do not waste your time attacking a purely psychic creature with a sword, they can be defeated by spells alone.

"Care must be taken when approaching any building; the swamps and forests are dangerous — make sure you have the experience to cope. Also, beware of water, you will be considerably weakened by the weight of your armour".

Alarian, bringing his stick to the ground raised himself out of his chair.

"I cannot promise you riches, though treasure there be in the Valley, I ask only that you save the Kingdom. Find the Amulet. Fill it with the six stones and you will have the ability to cheat death; to resurrect yourself within these safe castle walls. It will also prove invaluable in your search for the Helm of Evanna in the dark Lord's lair".

"Alas, I cannot help you much in your search for the Helm. For although I can again guide you to the areas of residual magick, the Helm, on its mistress' instruction, will not reveal itself to you unless you have at least the power of a Warlord. It will be up to you to build up this experience, I can only provide an occasional aura of magick to boost your powers, yet you will find that Vounim also has a way of watching over his followers and may surround you in a circle of evil.

"There is little time for discussion; I have arrived thirty years too late . . . and I fear Vounim knows it. The choice must be made here and it must be made now. Will you go and find the Helm of Evanna and bring it back here?"

Each of the five heads nodded as, in turn, Alarian gazed deeply into their eyes. Settling his hands on the carved walking stick at his side, the elderly wizard spoke to the assembled company in a low rumbling voice.

". . .who then will be first?"

The Valley is best described as a real-time adventure with graphics. It was developed as a direct result of reviewing and testing a large number of commercial offerings over Christmas 1980 and, we hope, overcomes many of the failings of these various alternatives.

The program has always been based on a modular system to simplify both its production and its documentation; this approach also means that the game can be tailored to suit the player's individual tastes. Again, because of the modular nature of the program it can easily be expanded provided you have more than the required 16K.

The published listing was developed and tested on a 32K

Commodore PET but will, if all non-essential spaces and REMs are removed, run in 16K. All graphics characters and cursor controls have been converted to CT's standards and character tables are provided. We are not suggesting that conversion to other systems is something that can be done in an evening but it is possible — we have versions running on Commodore 8032, TRS-80, Sharp MZ-80K, Sharp MZ-80A, BBC Model A and Model B, VIC-20 (16K), Dragon 32, Atari 400 and 800 (32K), ZX Spectrum (48K), Apple II (disc + 48K) with others on the way soon!

The best way to implement the program on your system is to key it in one module at a time following the notes. We have broken the listing down into the separate modules, each with a description of its main functions, to make this simpler. As each block is completed SAVE it on tape before adding the next; 16K is a lot of program to lose if you make a mistake!

PLAYING THE GAME

The objects of the game are explained in the introductory scenario, the actual mechanics of playing are described in the various sections that follow. Probably the best plan is to create a number of different characters, one of each type perhaps, and attempt to play each of them through the Valley and various scenes. The ultimate object of the game is to reach the highest rating level, 28, but along the way you will need to collect the various special treasures to ensure that if you are unfortunate enough to be killed you stand a chance of re-incarnation.

Game tactics are dependent on the type of characters you have chosen and are best developed by the player as the game progresses. A couple of hints may be welcome, however. If you are in combat with a monster that is stronger than you and are suffering great damage, then Spell 1 is possibly the best option to select. The other important tip is to remember that once you enter a scenario other than the Valley you are committed for a number of turns, so ensure that your stamina level is high.

However, before you can play the game you must enter the program so now is the time to start pressing keys!

INITIALISATION

Although it would be regarded as 'proper' to declare all the variables used at the start we have only initialised those necessary to begin the program correctly. Arrays are all DIMensioned to their correct sizes at this point and the vital positioning strings of cursor movements are also created.

The dummy READ routine between 300 and 320 moves the current position in DATA over the first block which will be used later for building castle-type scenes.

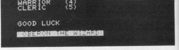
Monster data is loaded into three arrays; the monster name, its initial strength and its initial magical power. These starting values are modified according to the 'floor level' of the scene on which it appears. The monster details are presented in Table 1.

```
 99  REM ** DEFINE MAJOR VARIABLES
100  DIM D(3),G(73),P(8),N(8),S(4),T(2)
110  DIM M$(18),MS(18),N1(18)
120  VG$="":GC$="":F$="":DL$=""
130  TS=0:TN=0:TM=3:CF=0
140  D$="[HOM][21 CD]"
150  D1$=LEFT$(D$,17)
160  SP$="[39 SPC]"
170  R$="[30 CR]"
180  R1$=LEFT$(R$,21)
299  REM ** SKIP SCENE DATA
300  FOR I=1 TO 32
310  READ C$
320  NEXT I
329  REM ** LOAD MONSTER DATA
330  FOR I=0 TO 18
340  READ M$(I):READ MS(I):READ N1(I)
350  NEXT I
```

CHARACTER INITIALISATION

This block of program allows the user to set up his character with a name and one of a number of options of character type; Wizard, Cleric, Barbarian, etc. Table 2 contains information on the various character types.

Alternatively, if the game has been played before, the user may have a character stored on tape so the option exists to load this instead of starting afresh. The selection is made in 1050 after the name has been entered. The maximum length of name is 16 characters (checked in line 1040) and, because the string has to be entered as an INPUT, a simple bomb-proof trap is inserted at line 1030. This works by forcing an asterisk to appear under the cursor so if you simply press



```
CHARACTER TYPES...CHOOSE CAREFULLY

WIZARD     (1)
THINKER    (2)
BARBARIAN  (3)                KEY 1-5
WARRIOR    (4)
CLERIC     (5)

GOOD LUCK

 OBERON THE WIZARD
```

The first time you enter the game you can select your character type from the five available.

Monster	Physical	Magical	Code	Scenes
Thunder Lizard	50	0	V	V
Wolfen	9	0	A	V,W,S,C
Hob-Goblin	9	0	A	V,W,S,C
Orc	9	0	A	V,W,S,C
Ogre	23	0	A	V,W,S,C
Balrog	50	50	A	V,W,S,C
Fire Imp	7	3	E	V,W,C,L,T
Harpy	10	12	E	V,W,C,L,T
Fire Giant	26	20	E	V,W,C,L,T
Rock Troll	19	0	G	V,C
Centaur	18	14	H	V,W
Wyvern	36	12	F	W,S
Water Imp	15	15	L	W,S
Kraken	50	0	L	W,S
Minotaur	35	25	C	C,L,T
Wraith	0	30	C	C,L,T
Ring Wraith	0	45	C	C,L,T
Barrow Wight	0	25	B	L,T
Dragon	50	20	B	L,T

Table 1. The monsters, their strengths and their habitats. The code letter is extracted during the monster selection routine and matched against the current scene's 'allowable monsters' string. Note also that the two monsters coded with L can only be found in the lakes. The scenes are coded as: V-Valley, W-Woods, S-Swamps, C-Black Tower, L-Vounim's Lair and T-Temple of Y'Nagioth.

Return this is entered rather than nothing.

If data entry is from tape, the stamina value is set to maximum and the program then jumps to 1400. If, however, this is the first time through the game, the tape loading section is skipped and the player offered the choice of five character types. According to your selection, the initial values of your character's physical and magical strengths are determined together with your two 'gain' factors. These determine how much you gain from finds and how well various sections of the combat work. The P1 factor also acts as a limit on your character's growth; see Table 2.

Once the character has been selected, the Valley is then created for the first time and the initial status information displayed. The game now starts with your character located in the left-hand 'safe' castle and control is passed to the Movement routine.

```
 999  REM ** CHARACTER CHOICE AND LOAD
1000  PRINT "[CLS][CD]LOAD A CHARACTER FROM TAPE (Y/N) ?"
1010  VG$="YN":GOSUB 1500:REM ** UNIGET
1020  INPUT "[CD]CHARACTER'S NAME [2 CR]*[3 CL]";J$
1030  IF J$="*" THEN 1020
1040  IF LEN(J$)>16 THEN PRINT "[CD]TOO LONG":GOTO 1020
1050  IF GC$="N" THEN 1240
1060  PRINT "[CLS]PLACE DATA TAPE IN THE TAPE DECK"
1070  PRINT "[CD]IS IT REWOUND ?"
1080  GOSUB 1600:REM ** ANYKEY
1090  OPEN 1,1,0,J$
1100  INPUT#1,P$
1110  INPUT#1,T$
1120  INPUT#1,EX
1130  INPUT#1,TN
1140  INPUT#1,CS
1150  INPUT#1,PS
1160  INPUT#1,T(0)
1170  INPUT#1,T(1)
1180  INPUT#1,T(2)
1190  INPUT#1,C1
1200  INPUT#1,P1
1210  CLOSE 1
1220  C=150
1230  GOTO 1400
1240  PRINT "[CLS][2 CD]CHARACTER TYPES...CHOOSE
      CAREFULLY"
1250  PRINT
1260  PRINT "WIZARD     (1)"
1270  PRINT "THINKER    (2)"
1280  PRINT "BARBARIAN  (3)","KEY 1-5"
1290  PRINT "WARRIOR    (4)"
1300  PRINT "CLERIC     (5)"
1310  GET GC$:IF GC$="" THEN 1310
1320  A=VAL(GC$)
1330  IF A=1 THEN P$="WIZARD":P1=2:C1=0.5:CS=22:PS=28
1340  IF A=2 THEN P$="THINKER":P1=1.5:C1=0.75:CS=24:PS=26
1350  IF A=3 THEN P$="BARBARIAN":P1=0.5:C1=2:CS=28:PS=22
1360  IF A=4 THEN P$="WARRIOR":P1=1:C1=1.25:CS=26:PS=24
1370  IF A=5 THEN P$="CLERIC":P1=1.25:C1=1:CS=25:PS=25
1380  IF A<1 OR A>5 THEN P$="DOLT":P1=1:C1=1:CS=20:PS=20
1390  EX=5:C=150
1400  PRINT "[2 CD]GOOD LUCK"
1410  PRINT "[CD]";J$;" THE ";P$
1420  DF=150:DL$="D":GOSUB 36000:REM ** DELAY
1430  GOSUB 10000:REM ** VALLEY DRAW
1440  DF=5:GOSUB 36000:REM ** DELAY + UPDATE
1450  GOTO 2000:REM ** MOVEMENT
```

Type	P1	C1	CS	PS	C	CS Max	PS Max
Wizard	2.00	0.50	22	28	100	66	777
Thinker	1.50	0.75	24	26	113	72	241
Barbarian	0.50	2.00	28	22	125	77	89
Warrior	1.00	1.25	26	24	113	75	117
Cleric	1.25	1.00	25	25	113	74	157
Dolt	1.00	1.00	20	20	113	75	117

Table 2. The six possible character types with their initial values and the maximums to which their physical and magical strengths can rise.

FAST SUBROUTINES

UNIGET: This is a universal GET routine and is designed to operate in conjunction with the string VG$. It will only return to the main program if the character keyed is one of those in VG$.

ANYKEY: This routine is used in the tape save and load routines to allow the player to ensure the cassette is ready in the tape machine before proceeding; it can be removed or replaced as required.

COMBAT GET: A special timed GET routine for combat. It returns to the main program as soon as any key is pressed, assuming that this occurs within the time limit. The key pressed is held in GC$. If the time limit is exceeded the variable TV is set to 1. The routine also wipes

```
1499  REM ** UNIGET ROUTINE
1500  GET GC$:IF GC$="" THEN 1500
1510  FOR I=1 TO LEN(VG$)
1520  IF MID$(VG$,I,1)=GC$ THEN RETURN
1530  NEXT I
1540  GOTO 1500
1599  REM ** ANYKEY ROUTINE
1600  PRINT "[CD]** PRESS ANY KEY TO CONTINUE **"
1610  GET GC$:IF GC$="" THEN 1610
1620  RETURN
```

```
1699  REM ** COMBAT GET ROUTINE
1700  FOR I=1 TO 10:GET GC$:NEXT I:REM ** EMPTIES BUFFER
1710  TV=0
1720  FOR I=1 TO 60
1730  GET GC$:IF GC$="" THEN 1750
1740  GOTO 1770
1750  NEXT I
1760  TV=1:REM ** NO KEY PRESSED
1770  PRINT D$;SP$:REM ** WIPE AWAY MESSAGE
1780  RETURN
```

away the text message "* * *STRIKE QUICKLY * * *"

MOVEMENT

In many ways this represents the core of the whole program; it is certainly the most executed loop and controls access to all other major routines.

For such an important routine it occupies surprisingly little space, lines 2000 to 2260 in fact. Line 2000 is only used as an initial starting point when you first enter the Valley, either at the start of the game or when you return to the Valley from a scenario; all other calls are made to 2010. The POKE code 81 is the symbol used to display your current position in the Valley; Table 3 gives suggested alternatives for other systems.

The first operation is to give the character stamina a boost of 10, a dynamic refresh? Your current position is now examined to see if you are standing on a path or in the Valley and an appropriate message is printed requesting you to make a move. The player may move one square at a time in any direction; see Fig. 1. The choice of direction is made by keying one of the keys of the numeric keypad, the value of the key pressed then being inspected to find which direction it represents. In many programs of this type the checking is done by way of a look-up table which, although universal in operation and not restricted to numeric keypads is expensive in terms of memory.

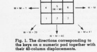
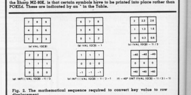


Fig. 1. The directions corresponding to the keys on a numeric pad together with their 40-column displacements.

Scene	Object	PET	MZ80-K	TRS-80
Valley	Border	214	166	191
	Safe Castle	219	74	35
	Path 'up'	78	118	154
	Path 'down'	77	119	169
	Woods	216	80	87
	Swamps	173	42	83
	Tower	87	65	84
	Character	81	202	79
Woods	Border	96	*	128
	Trees	88	70	91
	Lake	224	163	191
	Vounim's	230	239	86
	Character	81	202	79
Swamps	Border	96	*	128
	Tufts	45	227	45
	Lake	224	163	191
	Y'Nagioth	230	239	89
	Character	81	202	79
Tower Vounim & Y'Nagioth	Border	160	67	191
	Walls	160	67	191
	Stairs	102	109	153
	Doorway	104	212	176
	Treasures	42	107	42
	Character	81	202	79

Table 3. These are the recommended POKE codes for a selection of the systems that the program has been converted onto. One problem that may arise, shown here by the Sharp MZ-80K, is that certain symbols have to be printed into place rather than POKEd. These are indicated by an * in the Table.



(a) VAL (GC$) (b) VAL (GC$) – 1 (c) (VAL (GC$) – 1) / 3

(d) INT((VAL (GC$) –1) / 3 (e) INT ((VAL (GC$) – 1) / 3 –1 (f) – 40* (INT ((VAL (GC$) – 1) / 3) – 1)

Fig. 2. The mathematical sequence required to convert key value to row displacement.

Fortunately, one of the programming team had a mathematical background and produced the code between 2050 and 2090. Because each direction of movement corresponds directly to a screen displacement value, it must be possible to establish a simple mathematical relationship between the numeric key pressed and the direction in which you wish to move.

The routine starts at 2050 by clearing out the keyboard buffer, an essential operation to prevent old keystrokes causing problems. A charcter is now read in by the GET command in 2060 and checked to see if it is an 'E'. If it is an 'E', control is passed to the rating routine which gives your current 'Ego'. As we are only looking for numeric inputs in this routine we can test for validity by finding the VAL of the character; if this is 0 it must have been non-numeric so the program loops back for another character.

Once a valid key has been pressed, its value is held in the variable A and testing starts at line 2080. The first piece of code repetitively subtracts 3 from the value of A to determine the horizontal displacement. If keys 1,4 or 7 have been pressed we must wish to move left; 3,6 and 9 indicate a move ot the right and 2,5 and 8 maintain the current column position. As we are now left with a number between 1 and 3, we can determine the horizontal displacement by subtracting 2 from this remainder and this is done in 2090.

All we have to do now is to determine the correct vertical displacement, this also being computed in line 2090. Assuming a 40 column screen we must now establish the row we wish to move to; this is best explained by referring to Fig. 2. If we subtract 1 from the key value (Fig. 2b) and then divide by 3 (Fig. 2c) the INTeger part of the remaining number is related to both the key value and its corresponding row (Fig. 2d). Subtract 1 and multiply by − 40 and the resulting number is the vertical displacement.

Having computed the address of the position you wish to move to (held in variable W), we can now start to check what is in that position. These checks are preceded in line 2100 by incrementing the turns count and clearing away the movement message. The next block of lines inspects the contents of address W and is best shown as a small table:

Line	**Character**	**Action**
2110	Nothing	Jump to Movement
2120	Safe Castle	Jump to Quit routine
2130	Solid object	Try again!
2140	Scene code	Jump to Scene Control
2150	Scene exit	Jump to Scene Control
2160	Stairs	Jump to Stairs routine
2170	Lakes	Reverse the character code
2180	Special find	Jump to Special Finds

After all these checks have been performed it only remains to move your character into its selected position; line 2190 does this and because you stepped on a blank square, the program now generates a random number to see if there is either a hidden 'find' or a waiting monster. It does this in lines 2210 to 2230 and, depending on the value of the random number, control may be passed to either the Monster Selection routine or to the Finds routine.

If nothing is found a suitable message is printed and the program loops back to the beginning of movement at 2010. The DF value of 80 in line 2240 enables you to read the message; if you have stepped on the path, checked in line 2200, the delay is only set to 5 because there is no message to read.

```
1999  REM ** MOVEMENT ROUTINE
2000  M=W:PK=PEEK(W):POKE M,81
2010  C=C+10
2020  IF PK=77 OR PK=78 THEN 2040
2030  PRINT D$;"YOUR MOVE...WHICH DIRECTION ?":GOTO 2050
2040  PRINT D$;"SAFE ON THE PATH...WHICH WAY ?"
2050  FOR I=1 TO 10:GET GC$:NEXT I:REM ** CLEAR KBD BUFFER
2060  GET GC$:IF GC$="E" THEN 45000:REM ** EGO
2069  REM ** SPECIAL ROUTINE FOR NUMERIC KEYPADS
2070  A=VAL(GC$):IF A=0 THEN 2060
2080  IF A>3 THEN A=A-3:GOTO 2080
2090  W=M+A-2-40*(INT((VAL(GC$)-1)/3)-1)
2100  TN=TN+1:PRINT D$;SP$
2109  REM ** AM I STEPPING ON SOMETHING ?
2110  Q=81:Q1=PEEK(W):IF Q1=32 OR Q1=45 THEN 2190
2120  IF Q1=219 THEN 40000:REM ** QUIT
2130  IF Q1=214 OR Q1=160 OR Q1=88 THEN TN=TN-1:GOTO
      2030:REM ** HIT WALL OR TREE
2140  IF Q1=216 OR Q1=87 OR Q1=173 OR Q1=230 THEN 9000:
      REM ** SCENE ENTRY
2150  IF Q1=104 OR Q1=96 THEN 9090:REM ** SCENE EXIT
2160  IF Q1=102 THEN 15000:REM ** STAIRS
2170  IF Q1=224 OR (GC$="5" AND PK=224) THEN Q=209:
      C=C-20:IF C<=0 THEN 55000:REM ** WATER
2180  IF Q1=42 THEN 2800:REM ** SPECIAL FIND
2190  POKE M,PK:PK=PEEK(W):M=W:POKE M,Q
2200  IF PK=77 OR PK=78 THEN DF=5:GOTO 2250
2209  REM ** NOTHING, MONSTER OR FIND PERHAPS ?
2210  RF=RND(TI)
2220  IF RF<0.4 THEN 3000:REM ** MONSTER SELECT
2230  IF RF>0.8 THEN 2300:REM ** FIND SELECT
2240  PRINT D$;"NOTHING OF VALUE...SEARCH ON":DF=80
2250  GOSUB 36000:REM ** DELAY + UPDATE
2260  GOTO 2010
```

FINDS

The 'ordinary finds' module starts lines 2300-2310 by randomly selecting one of four finds — three good, one not so good. A random integer between 1 and 6 is generated and two line numbers appear twice in the ON . . . GOSUB list, thus giving probabilities of roughly 16%, 32%, 32% and 16% to the finds.

The first find, starting at 2340 is the bad one. Although line 2350 boosts combat strength by an amount dependent on FL, magical ability drops (FL again being a factor) and stamina is reduced by 20. Line 2360 jumps to Death routine if C falls below zero.

At line 2380 'a hoard of gold' is found. Treasure is incremented by a value between

```
2299  REM ** FINDS ROUTINE
2300  RF=INT(RND(TI)*6+1)
2310  ON RF GOSUB 2340,2380,2380,2410,2410,2440
2320  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
2330  GOTO 2010
2340  PRINT D$;"A CIRCLE OF EVIL...DEPART IN HASTE !"
2350  CS=CS+INT((FL+1)/2):PS=PS-INT((FL+1)/2):C=C-20
2360  IF C<=0 THEN 55000:REM ** DEATH
2370  RETURN
2380  PRINT D$;"A HOARD OF GOLD"
2390  TS=TS+INT(FL*RND(TI)*100+100)
2400  RETURN
2410  PRINT D$;"YOU FEEL THE AURA OF THE DEEP MAGIC..."
2420  PRINT "[8 SPC]...ALL AROUND YOU..."
2430  GOTO 2450
2440  PRINT D$;"...A PLACE OF ANCIENT POWER..."
2450  PS=PS+2+INT(FL*P1):CS=CS+1+INT(FL*C1):C=C+25
2460  RETURN
```

100 and 700, depending on FL and a random factor. The third and fourth finds are not monetary but physical and magical; although different messages are printed, lines 2410 and 2440.

Each of the four subroutines returns to line 2320 for a delay and update before control is returned to the Movement routine, line 2010.

SPECIAL FINDS

If the Movement routine establishes that you have stepped onto an asterisk, a jump is made to the Special Finds module at line 2800. Here you are placed on the marker which is then erased (PK = 32); once you've picked up a special find, it's gone for good! Next, a random number is generated to decide what you've found; the 'Movement' message is wiped and a series of tests begins.

The first test (line 2820) succeeds if you're in Vounim's Lair, S=6, have a full Amulet, T(1)=6, a rating greater than 25, and have not already found the Helm of Evanna, T(2)=0. You also have to very lucky, RN>0.95!

Line 2830 tests for the empty Amulet; this can only be found in the Temple of Y'Nagioth, S=5 with RN >0.85. You may only have one Amulet at a time, T(0)=0. Note that although you need a full Amulet to obtain the Helm, losing the Amulet later (through reincarnation) means you are free to find another one; no problems arise if you already have the Helm.

The next line, 2840, checks for Amulet stones which only occur in the Black Tower, S=4. Not only must you first have the Amulet, T(0)=1, and space left in it, T(1)<6, but you must be on a sufficiently high floor. The first stones can be found low down the Tower but as you find each stone you must venture higher to find the remaining ones, FL>T(1). Assuming you have found an Amulet stone, lines 2910-2920 decide whether it fits or not. Since RN must already be greater than 0.7 to get to these lines, the condition that RN>0.85 here gives a 50-50 chance of the stone being the right one.

If the tests in lines 2820-2840 all fail then you have found either a precious stone or a worthless bauble. The random factor of 0.43 in line 2850 was chosen to get a long-term average of roughly 50% between precious stones and worthless baubles since RN may have been 'filtered' by the previous tests. For example, if all three tests failed on factors other than the value of RN, it could be anything between 0 and 1 on line 2850 and the probability is 43% that you have a worthless bauble. On the other hand, if line 2840 failed only because RN<0.7 then the probability shifts to 0.43/0.7=61.4%.

As well as obtaining the objects themselves your treasure, TS, is updated in line 2930 by an amount which depends on the number of items you've already found. Baubles and wrong Amulet stones don't count and bypass this line.

```
2799  REM ** SPECIAL FINDS ROUTINE
2800  POKE M,32:M=W:PK=32:POKE M,81
2810  RN=RND(TI):PRINT D$;SP$
2820  IF S=6 AND RN>O.95 AND T(1)=6 AND T(2)=0 AND RT>25
      THEN T(2)=1:GOTO 2870
2830  IF S=5 AND RN>0.85 AND T(0)=0 THEN T(0)=1:GOTO 2880
2840  IF S=4 AND RN>0.7 AND T(0)=1 AND T(1)<6 AND FL>T(1)
      THEN 2890
2850  IF RN>0.43 THEN PRINT D$;"A WORTHLESS BAUBLE":
      GOTO 2940
2860  PRINT D$;"A PRECIOUS STONE :":GOTO 2930
2870  PRINT D$;"YOU FIND THE HELM OF EVANNA !":GOTO 2930
2880  PRINT D$;"THE AMULET OF ALARIAN...EMPTY...":
      GOTO 2930
2890  PRINT D$;"AN AMULET STONE..."
2900  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
2910  IF RN>0.85 THEN PRINT "[CD]...BUT THE WRONG ONE !":
      GOTO 2940
2920  PRINT "[CD]...THE STONE FITS !":T(1)=T(1)+1
2930  TS=TS+100*(T(0)+T(1)+T(2)+FL)
2940  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
2950  GOTO 2010
```

MONSTER SELECTION

If you have the misfortune to draw a monster at the end of the Movement routine, control is passed to the segment of code which starts at 3000. A random number is generated between 1 and 16 and tested to see if it is greater than 9. This test is made to ensure that the stronger monsters cannot occur too frequently; the checks and limits for this are established in line 3020.

If the character is currently standing or swimming in a lake the choice of monsters is limited by line 3030 to the Water Imp and the Kraken, both prefixed L in the DATA.

The most unpleasant general monster is the Balrog and if he is drawn from the array, a further check is made in line 3040 to ensure that he appears less frequently.

Some monsters live only in the rarified heights of the Black Tower or one of the two special castle-type scenes and if these are drawn from the array, a further check is made in line 3050 to ensure that these conditions are met.

Once an acceptable monster has been selected from the array, the left-hand character of its name is stripped off to see if it can exist in the current scenario; this character is then checked against F$ in lines 3060 to 3090. If all is correct, the name of the chosen beast is displayed on the screen and combat commences. The base strengths of each monster are held in arrays MS() and N1() and these values are further modified by the code between lines 3120 and 3170 to produce the actual strengths of the chosen monster.

```
2999  REM ** MONSTER SELECTION ROUTINE
3000  PRINT D$;"** BEWARE...THOU HAST ENCOUNTERED **"
3010  MS=0:N=0:CF=1
3020  RF=INT(RND(TI)*17):IF RF>9 AND RND(TI)>0.85
      THEN 3020
3030  IF Q1=224 OR PK=224 THEN RF=INT(RND(TI)*2+17)
3040  IF RF=16 AND RND(TI)<0.7 THEN 3020
3050  IF FL<5 AND RF=15 THEN 3020
3060  X$=LEFT$(M$(RF),1)
3070  FOR I=1 TO LEN(F$)
3080  IF MID$(F$,I,1)=X$ THEN 3110
3090  NEXT I
3100  GOTO 3020
3110  M$=RIGHT$(M$(RF),LEN(M$(RF))-1)
3120  IF MS(RF)=0 THEN 3150
3130  MS=INT((CS*0.3)+MS(RF)*FL^0.2/(RND(TI)+1))
3140  IF N1(RF)=0 THEN 3160
3150  N=INT(N1(RF)*FL^0.2/(RND(TI)+1))
3160  U=INT((RF+1)*(FL^1.5))
3170  IF RF>23 THEN U=INT((RF-22)*FL^1.5)
3180  PRINT "[CD]";LEFT$(R$,12-(LEN(M$))/2);"AN EVIL ";M$
3190  DF=40:GOSUB 36000:REM ** DELAY + UPDATE
```

CHARACTER'S COMBAT

The action of fighting a monster can be broken down into three main sections; you hitting it, it hitting you and you casting a spell. The Spells are controlled by their own section of code that will be described later and can be simply treated as a jump out of the physical combat routines.

The Character's Combat section is located from 3570 to 3910 but before this can be executed we must determine whether you have surprised the beast or not. This is tested for in line 3500 where a random number is generated giving a 60/40 chance of you surprising the monster. If you do have surprise you are then offered the option of retreating from combat. The "R" key must be pressed within the time limit of the Combat Get routine or control passes directly to the Monster's Combat.

If you do choose to retreat a suitable message is displayed and the program goes back to the Movement routine. Choosing to attack, the message " * * * Strike Quickly * * *" is displayed and you have the choice of attacking its Head, Body or Limbs with the further option of trying to cast a Spell. If no key was pressed or the wrong one was chosen, control passes to the Monster's Combat via a suitable message. The control for this section of the combat is handled by lines 3570 to 3610.

Assuming that you have pressed a valid key the program checks in line 3630 to see if you wished to cast a spell and if so passes control to the Spell Control section. Before determining how much damage, if any, you have done to the beast, the program computes your current experience factor (in line 3620) and deducts one stamina point. If you have exhausted yourself attempting to fight, the program detects this in 3660 and passes control to the Death routine.

Because each of the three target areas of the monster have different levels of vulnerability the code between 3670 and 3710 determines whether you hit the beast or not and, if you did, sets the damage factor variable Z to the appropriate level. As it is possible to strike a heavy blow which will leave the monster helpless we must first inspect the corresponding flag, HF, which tells us if the beast is certain to die on this attempt. If this flag has not been set in line 3750, we calculate the damage done to the monster and display it — it is possible to hit the monster yet do no damage! There are now several options available to us and these are sorted out by the rest of the routine from 3800 to 3910. The first alternative is that we have killed it, in which case we collect experience, reset the combat flags and go back to the movement routine (lines 3860 to 3890). Our second option is that we have done so much damage to the beast that it is unable to have another go at us. In this case we set the flag, HF, and go back to the " * * * Strike Quickly * * *" message. The remaining alternatives are that we either did no damage or insufficient to cripple the monster and in both cases, control now passes to the Monster's Combat routine.

```
3499  REM ** CHARACTER'S COMBAT ROUTINE
3500  IF RND(TI)<0.6 THEN 4000:REM ** MONSTER'S COMBAT
3510  PRINT D$;"YOU HAVE SUPRISE...ATTACK OR RETREAT"
3520  GOSUB 1700:REM ** COMBAT GET
3530  IF GC$="R" THEN 3900
3540  IF TV=1 THEN 3600
3550  IF GC$<>"A" THEN 4000
3560  DF=30:DL$="D":GOSUB 36000:REM ** DELAY
3570  PRINT D$;"*** STRIKE QUICKLY ***"
3580  GOSUB 1700:REM ** COMBAT GET
3590  IF TV=0 THEN 3620
3600  PRINT D$;"* TOO SLOW...TOO SLOW *"
3610  HF=0:GOTO 3830
3620  E=39*LOG(EX)/3.14
3630  IF GC$="S" THEN 4500:REM ** SPELL CONTROL
3640  IF MS=0 THEN PRINT D$;"YOUR SWORD AVAILS YOU NOUGHT
      HERE":GOTO 3830
3650  C=C-1
3660  IF C<=0 THEN PRINT D$;"YOU FATALLY EXHAUST YOURSELF":
      GOTO 55000:REM ** DEATH
3670  RF=RND(TI)*10
3680  IF GC$="H" AND (RF<5 OR CS>MS*4) THEN Z=2:GOTO 3730
3690  IF GC$="B" AND (RF<7 OR CS>MS*4) THEN Z=1:GOTO 3730
3700  IF GC$="L" AND (RF<9 OR CS>MS*4) THEN Z=0.3:
      GOTO 3730
3710  PRINT D$;"YOU MISSED IT !"
3720  HF=0:GOTO 3830
3730  IF HF=1 THEN D=MS+INT(RND(TI)*9):HF=0:GOTO 3760
3740  D=INT((((CS*50*RND(TI))-(10*MS)+E)/100)*Z):IF D<0
      THEN D=0
3750  IF CS>(MS-D)*4 THEN HF=1
3760  MS=MS-D
3770  PRINT D$;"A HIT..."
3780  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
3790  IF D=0 THEN PRINT D$;"[8 CR]BUT...NO DAMAGE":HF=0:
      GOTO 3830
3800  PRINT D$;"[8 CR]";D;" DAMAGE...":IF MS<=0 THEN
      3860:REM ** IT'S DEAD
3810  IF HF=1 THEN DF=30:DL$="D":GOSUB 36000:REM ** DELAY
3820  IF HF=1 THEN PRINT "[CD]THE ";M$;" STAGGERS
      DEFEATED"
3830  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
3840  IF HF=1 THEN 3570
3850  GOTO 4000:REM ** MONSTER'S COMBAT
3860  PRINT D$;"[2 CD]...KILLING THE MONSTER..."
3870  EX=EX+U:HF=0:CF=0
3880  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
3890  GOTO 2010:REM ** MOVEMENT
3900  PRINT D$;"KNAVISH COWARD !":CF=0
3910  GOTO 3880
```

MONSTER'S COMBAT

Unlike the character, the monster has only two possible methods of attack. Its normal approach is to hit you but as magical monsters, ones with no physical strength, are unable to wield swords and the like, there is the option to attack you with a lightning bolt. To make life even more unpleasant this option is extended to a physical monster whose strength has fallen below its psi power!

The options for the monster are checked at the beginning of its routine, lines 4000 to 4040, before a random number is generated in line 4050 determining the outcome. The random number can be between 1 and 10 and there are eight possible messages, two messages appearing twice.

Depending on the value of the random number the appropriate message is selected by line 4060 and control is passed to the appropriate section of the routine.

If the monster has missed you or used up all its stamina in the attempt, the section between 4240 and 4290 takes the program back to the Character's Combat or the Movement routine respectively.

Just as your character can hit a selected area of the beast, the reverse is now true but the area is selected randomly by line 4060. Again, as in the Character's Combat, a damage factor Z is set to an appropriate value and the potential damage done to you G is calculated by line 4160. This amount is deducted from your character's stamina by line 4180 and your health is then examined by line 4220; if you are now an ex-character, control is passed to the Death routine.

As mentioned earlier in this section, the monster can throw a lightning bolt at you and if this option is selected by line 4030, control is passed to the section of program from 4300 to 4410. This computes the possiblity of the lightning bolt hitting or missing and, if it hits, how much damage will result.

The outcome of the Monster's Combat routine is again a three-way option; if it has killed you so the game jumps to Death; if it has wounded but not killed (or missed completely) so control passes back to Character's Combat; or it has killed itself in which case control passes back to the Movement section.

```
3999  REM ** MONSTER'S COMBAT ROUTINE
4000  PRINT D$;"THE CREATURE ATTACKS..."
4010  DF=50:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
4020  IF MS=0 THEN 4300:REM ** PSIONIC ATTACK
4030  IF MS<N AND N>6 AND RND(TI)<0.5 THEN 4300
4040  MS=MS-1:IF MS<=0 THEN 4240
4050  RF=INT(RND(TI)*10+1)
4060  ON RF GOTO 4070,4080,4090,4100,4110,4110,4120,4120,
      4130,4140
4070  PRINT D$;"IT SWINGS AT YOU...AND MISSES":GOTO 4280
4080  PRINT D$;"YOUR BLADE DEFLECTS THE BLOW":GOTO 4280
4090  PRINT D$;"...BUT HESITATES, UNSURE...":GOTO 4280
4100  Z=3:PRINT D$;"IT STRIKES YOUR HEAD !":GOTO 4150
4110  Z=1.5:PRINT D$;"YOUR CHEST IS STRUCK !":GOTO 4150
4120  Z=1:PRINT D$;"A STRIKE TO YOUR SWORDARM !":
      GOTO 4150
4130  Z=1.3:PRINT D$;"A BLOW TO YOUR BODY !":GOTO 4150
4140  Z=0.5:PRINT D$;"IT CATCHES YOUR LEGS !"
4150  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
4160  G=INT((((MS*75*RND(TI))-(10*CS)-E)/100)*Z)
4170  IF G<=0 THEN G=0:PRINT D$;"...SAVED BY YOUR
      ARMOUR ![2 SPC]":GOTO 4280
4180  C=C-G
4190  IF G>9 THEN CS=INT(CS-G/6)
4200  IF G=0 THEN PRINT D$;"SHAKEN......BUT NO DAMAGE
      DONE":GOTO 4280
4210  PRINT D$;"YOU TAKE...[6 SPC][6 CL]";G;" DAMAGE...
      [6 SPC]"
4220  IF CS<=0 OR C<=0 THEN 55000:REM ** DEATH
4230  GOTO 4280
4240  PRINT D$;"...USING ITS LAST ENERGY IN THE ATTEMPT"
4250  EX=INT(EX+U/2):CF=0
4260  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4270  GOTO 2010:REM ** MOVEMENT
4280  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4290  GOTO 3570:REM ** CHARACTER'S COMBAT
4299  REM ** MONSTER'S PSIONIC ATTACK
4300  PRINT D$;"...HURLING A LIGHTNING BOLT AT YOU !"
4310  G=INT(((100*N*RND(TI))-(PS+E))/100):N=N-5:IF G>9
      THEN N=N-INT(G/5)
4320  DF=80:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
4330  IF N<=0 THEN N=0:GOTO 4240
4340  IF RND(TI)<0.25 THEN 4410
4350  IF G<=0 THEN G=0:GOTO 4400
4360  PRINT D$;"IT STRIKES HOME !"
4370  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
4380  C=C-G:IF G>9 THEN PS=INT(PS-G/4)
4390  GOTO 4210
4400  PRINT D$;"YOUR PSI SHIELD PROTECTS YOU":GOTO 4280
4410  PRINT D$;"...MISSED YOU !":GOTO 4280
```

SPELL CONTROL

If the option of using a spell is chosen during the Combat routine, control jumps to the routine located from 4500. An initial message is displayed asking which spell you wish to cast and the player's reply is collected by the Combat Get routine.

If you don't press a key within the allotted time the program jumps back to the "Too Slow" message and you have missed your chance; this test is performed in line 4510.

As there are only three spells available in the 16K version (we've left plenty of room for expansion) a check is made at line 4520 to ensure that the key you pressed is valid and if not, a suitable message is printed and control passes back to the Combat routine through line 4640.

Given that you have pressed a valid key, line 4540 now checks to see if you are strong enough to use the chosen spell; if not, control jumps to line 4590 with a suitable message and, once again, the program goes back to the Combat routine via line 4640.

If you meet the requirements to use the spell the appropriate subroutine is selected by line 4550 and control passes to the chosen spell subroutine.

On RETURNing from the spell subroutine a flag variable, SC, will have been given a value between 1 and 7 and this value represents the outcome of casting the spell. Line 4560 causes the program to jump to the correct message and result.

```
4499  REM ** SPELL CONTROL ROUTINE
4500  PRINT D$;"WHICH SPELL SEEK YE ? ":GOSUB 1700:REM **
      COMBAT GET
4510  IF TV=1 THEN 3600:REM ** TOO SLOW
4520  IF VAL(GC$)>0 AND VAL(GC$)=<3 THEN 4540
4530  PRINT D$;"NO SUCH SPELL...[5 SPC]":GOTO 4640
4540  IF 4*PS*RND(TI)<=N THEN 4590
4550  ON VAL(GC$) GOSUB 5000,5200,5400
4559  REM ** SC CONTAINS OUTCOME FLAG
4560  ON SC GOTO 4620,4640,4660,4570,4600,4580,4590
4570  PRINT D$;"IT IS BEYOND YOU[5 SPC]":GOTO 4640
4580  PRINT "BUT THE SPELL FAILS...!":GOTO 4640
4590  PRINT D$;"NO USE, THE BEAST'S PSI SHIELDS IT":
      GOTO 4640
4600  PRINT D$;"THE SPELL SAPS ALL YOUR STRENGTH"
4610  GOTO 55000:REM ** DEATH
4620  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4630  GOTO 2010:REM ** MOVEMENT
4640  DF=60:GOSUB 36000:REM ** DELAY + UPDATE
4650  GOTO 4000:REM ** MONSTER'S COMBAT
4660  DF=60:GOSUB 36000:REM ** DELAY + UPDATE
4670  GOTO 3570:REM ** CHARACTER'S COMBAT
```

THE SPELLS

SLEEPIT: The first checks made in this subroutine are in line 5000 where stamina is deducted and, if your total has dropped below 0, the program is RETURNed via line 4560 to the Death routine. If your stamina is still healthy the message for the spell is printed by lines 5010 to 5050, moving on to line 5060 to see if the spell actually worked.

The possiblity of the spell working is 50% and if successful the program prints the welcome message and updates your experience before going back to the Movement routine via line 4560. If it fails the program still RETURNs via 4560 but the value of the SC flag passes control to the Monster's Combat.

PSI-LANCE: An initial check is made in line 5200 to see if the character has met the requirements for the spell to function, if not the program RETURNs to 4560 and then to the Monster's Combat after displaying a suitable message.

Once again the cost of the character in stamina is deducted, this time in line 5210, and the check made to see if he has exhausted himself is done. If you are still alive and fighting, the check is made in line 5220 to see if you are really attacking a monster with some psi power, if not then you are wasting your time so control passes back via 4560 to the Monster's Combat.

If all is well at this stage the text for the spell is printed by lines 5230 and 5240 and the test to see if the spell was successful is then made at line 5250. If the spell failed the program RETURNs to 4560 and the Monster's Combat after printing a suitable message.

The amount of damage your spell did to the monster is calculated in 5260 and if no damage was inflicted the program goes back to 4560 and the Monster's Combat. If you did damage the creature the amount inflicted is displayed and deducted from the monster's strength and psi power; these values are then checked to see if the monster has expired by line 5310. If the monster is still alive the program returns to Monster's Combat via 4560; otherwise line 5320 tells you that you have killed the beast, your experience is subsequently updated by line 5330 and the program RETURNs to 4560 and then back to the Movement routine.

CRISPIT: Once again an initial test is made to see if your character meets the requirements to use the spell, if not, it's back to line 4560 and then to the Monster's Combat. If you can use the spell line 5410 tests to see if the stamina you use to cast the spell has killed you off; if it has it passes control to Death via 4560.

The spell message is printed by lines 5420 and 5470 before line 5480 computes the outcome of the spell. Once again, if the spell failed it RETURNs to 4560 and then to the Monster's Combat.

If the spell worked, line 5490 calculates the damage done; if none, then it's back to 4560 and the Monster's Combat again! Given the spell has worked, the damage is deducted from the monster's physical strength and if it hasn't got any it is deducted from its psi power; all this is handled by lines 5510 to 5530. In fact, if you do more than a certain amount of damage, line 5520 takes some extra points off the psi power as well.

The rest of the routine is concerned with printing out the amount of damage done and checking to see if the monster is now dead. Depending on the result of these tests the program can jump to either the Movement routine or the Monster's Combat via the ever-faithful 4560.

```
4999  REM ** SPELL 1 (SLEEPIT)
5000  C=C-5:IF C<=0 THEN SC=5:RETURN
5010  PRINT D$;"SLEEP YOU FOUL FIEND THAT I MAY ESCAPE"
5020  PRINT "AND PRESERVE MY MISERABLE SKIN"
5030  DF=180:GOSUB 36000:REM ** DELAY + UPDATE
5040  PRINT D$;"THE CREATURE STAGGERS..."
5050  DF=40:DL$="D":GOSUB 36000:REM ** DELAY
5060  IF RND(TI)<0.5 THEN 5090
5070  PRINT "AND COLLAPSES...STUNNED"
5080  EX=INT(EX+U/2):CF=0:SC=1:RETURN
5090  PRINT "BUT RECOVERS WITH A SNARL !"
5100  SC=2:RETURN
5199  REM ** SPELL 2 (PSI-LANCE)
5200  IF MS>C OR PS<49 OR EX<1000 THEN SC=4:RETURN
5210  C=C-10:IF C<=0 THEN SC=5:RETURN
5220  IF N=0 THEN PRINT D$;"THIS BEAST HAS NO PSI TO
      ATTACK":SC=2:RETURN
5230  PRINT D$;"WITH MY MIND I BATTLE THEE FOR MY LIFE"
5240  DF=120:GOSUB 36000:REM ** DELAY + UPDATE
5250  RF=RND(TI):IF RF<0.4 AND N>10 THEN SC=6:RETURN
5260  D=INT((((CS*50*RF)-5*(MS+N)*E)/50)/4)
5270  IF D<=0 THEN D=0:SC=7:RETURN
5280  PRINT D$;"THE PSI-LANCE CAUSES ";D*2;" DAMAGE"
5290  N=N-3*D:IF N<=0 THEN N=0
5300  MS=MS-D:IF MS<=0 THEN MS=0
5310  IF (MS+N)>0 THEN SC=2:RETURN
5320  PRINT "[CD]...KILLING THE CREATURE"
5330  EX=EX+U:CF=0:SC=1:RETURN
5399  REM ** SPELL 3 (CRISPIT)
5400  IF PS<77 OR EX<5000 THEN SC=4:RETURN
5410  C=C-20:IF C<=0 THEN SC=5:RETURN
5420  PRINT D$;"WITH THE MIGHT OF MY SWORD I SMITE THEE"
5430  PRINT "WITH THE POWER OF MY SPELL I CURSE THEE"
5440  PRINT "BURN YE SPAWN OF HELL AND SUFFER..."
5450  DF=240:GOSUB 36000:REM ** DELAY + UPDATE
5460  PRINT D$;"A BOLT OF ENERGY LASHES AT THE BEAST..."
```

```
5470  DF=80:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
5480  IF RND(TI)>(PS/780)*(5-P1) THEN PRINT D$;"MISSED
      IT !":SC=2:RETURN
5490  D=INT((CS+PS*RND(TI))-(10*N*RND(TI)))
5500  IF D<=0 THEN D=0:SC=7:RETURN
5510  IF MS=0 THEN N=N-D:GOTO 5530
5520  MS=MS-D:IF D>10 THEN N=INT(N-(D/3))
5530  PRINT D$;"IT STRIKES HOME CAUSING ";D;" DAMAGE
      [2 SPC]!"
5540  IF (MS+N)<=0 THEN 5570
5550  DF=80:DL$="D":GOSUB 36000:REM ** DELAY
5560  SC=2:RETURN
5570  PRINT "[CD]THE BEAST DIES SCREAMING !"
5580  EX=EX+U:CF=0:SC=1:RETURN
```

SCENE CONTROL

Having drawn the Valley, the Path and the scenario positions, our hero is now free to wander where he chooses. A monster may kill him but with a little luck and fast reflexes, he will sooner or later enter one of the scenarios, or having entered may wish to escape!

To do this he simply moves onto the scenario or exit symbol. The part of the program concerned with movement, lines 2000-2260, detects the symbol and the program is directed to Scene Control at line 9000 or Scene Exit at line 9090.

A check is made at line 9000 to see if you are attempting to enter one of the secondary scenarios direct from the water, POKE code 224. If you are, the program is directed back to Movement via line 9110 which inhibits the turn count and the stamina refresh.

The two scene building arrays, P() and N(), are dealt with in lines 9010-9050. Array P() is used to fix the pattern of rooms on different levels of a scenario and if the scenario had more than one level, each level retains the same room pattern while you remain in that scenario. Array P() has all previous values zeroed.

Array N() has a series of random numbers, integers between 4 and 8, assigned to it (5 is not permitted as it can produce an unacceptable pattern). This array determines the depth of the rooms in the primary and secondary castle-type scenarios, Vounim's Lair and the Temple of Y'Nagioth (pronounced Ee-nag-ee-oth).

Entering a scenario from the Valley, tested for in line 9060, MP is assigned your last position in the Valley, M, so that when you leave the scenario you will return to the same position in the Valley.

A random integer between 1 and 30 is generated in line 9070 and is assigned to P(2). This is used in all scenarios to determine the position of lakes or the patterns of rooms. Line 9080 sets a temporary variable TF to the number of turns you have had so far, TN. This is then used when you try to exit from a scenario testing whether you been 'in' for a certain number of turns. If you have, this directs the program past the initial portion of the 'scene exit' to line 9130.

Moving on to line 9090, a test is made to see if you have remained within a scenario for a minimum number of turns. Thus if the turns count is equal to or larger than the generated random number, the program jumps to line 9130. If not you are barred from exiting the scenario and line 9110 inhibits the turns count and the stamina refresh. The program is then directed back to the Movement routine at line 9120.

With the Valley still displayed on the screen, line 9130 POKEs a space into your present position and POKEs your character symbol to the position of the scenario symbol. W is the position you are about to go to . . .

The lines 9140-9220 check to see what scenario or exit symbol has been stepped on thus determining the scene change required; Q1 represents the POKE code of position W.

If the symbol which frames the Swamps or Woods is found, a shifted space (POKE code 96) as specified in line 9140, then you must be leaving the scene and entering the Valley, S = 1:FL = 1. Line 9150 detects gateways, Q1 = 104, in the Black Tower, S = 4, denoting that you are leaving the Tower and entering the Valley.

If a gateway is detected in the Temple, S = 5, or the Lair, S = 6, line 9160 directs the program to allow you to enter the Swamps, S = 2, or the Woods, S = 3, respectively. These secondary scenarios are only found on a lake in these scenarios but in both cases the scene number of the Swamps or Woods is 3 below their secondary scenarios, S = S − 3. FL is determined in a similar manner, FL = FL − 4, and your position in the Swamps or Woods is reset to your original position outside the secondary scenario, M = MW.

A check is made at line 9170 to see if you have entered the Swamps, setting the scene number, S = 2, and the level, FL = 2, if you have 9180 does the check for the Woods.

Another check is made at line 9190 for either Swamps or Woods to assign two string variables, D2$ and R2$, if either of these scenarios are entered. Both D2$ and R2$ are cursor control movements: D2$ being a random number of Cursor Downs between 0 and 9 and R2$ being a random number of Cursor Rights between 1 and 30; this is the first use of P(2).

In line 9210, there is a check of one of the secondary scenarios in the Swamps or the Woods. If the Lair or Temple symbol, POKE code 230 is recognised, the appropriate scene number is assigned, S = S + 3 and level number, FL = FL + 4. The temporary position variable, MW, is assigned your position, M, immediately prior to entering the secondary scenario.

The program is directed to the appropriate subroutine in line 9220 and then in line 9230, the program is directed to the delay and status update subroutine. On RETURNing, line 9240 takes you back to the Movement routine and the new scenario is displayed awaiting further exploration.

```
8999  REM ** SCENARIO CONTROL ROUTINE
9000  IF Q1=230 AND PK=224 THEN PRINT D$;"YOU CANNOT
      ENTER THIS WAY...":GOTO 9110
9010  FOR I=2 TO 7
9020  P(I)=0
9030  N(I)=INT(RND(TI)*5+4)
9040  IF N(I)=5 THEN 9030
9050  NEXT I
9060  IF S=1 THEN MP=M
9070  P(2)=INT(RND(TI)*30+1)
9080  TF=TN:GOTO 9130
9089  REM ** EXIT FROM SCENARIO
9090  IF TN>TF+INT(RND(TI)*6+1) THEN 9130
9100  PRINT D$;"THE WAY IS BARRED"
9110  TN=TN-1:C=C-10:DF=100:DL$="W":GOSUB 36000:
      REM ** DELAY + WIPE
9120  GOTO 2010
9130  C=C-10:POKE M,32:POKE W,Q
9140  IF Q1=96 THEN S=1:FL=1
9150  IF Q1=104 AND S=4 THEN S=1:FL=1
9160  IF Q1=104 AND S=5 OR S=6 THEN S=S-3:FL=FL-4:M=MW
9170  IF Q1=173 THEN S=2:FL=2
9180  IF Q1=216 THEN S=3:FL=3
9190  IF Q1=216 OR Q1=173 THEN D2$=LEFT$(D$,INT(RND(TI)*
      10)):R2$=LEFT$(R$,P(2))
9200  IF Q1=87 THEN S=4:FL=2
9210  IF Q1=230 THEN S=S+3:FL=FL+4:MW=M
9220  ON S GOSUB 10000,12000,12010,14000,14010,14010
9230  DF=5:GOSUB 36000:REM ** DELAY + UPDATE
9240  GOTO 2000:REM ** MOVEMENT
```

THE VALLEY

Let us start where our 'alter ego' whatever his character type, will step out into this adventure . . . the Valley. Line 10000 clears the screen sets F$ to "VAEGH" which determines what monsters may be found in the Valley, ie monsters from groups V,A,E,G, and H (see Table 1). The difficulty level is set to 1 (FL has a bearing on how strong the monsters are) and finally sets the scene number also to 1 . . . the Valley.

First let us draw the bounds of the Valley; this consists of a rectangle 39 characters wide by 14 characters high. Lines 10010 and 10050 draw in the top and bottom frames of the Valley, between them a FOR . . . NEXT loop draws the vertical frames consisting of the appropriate characters separated by 37 spaces. These are drawn 12 times giving an internal playing area of 37 by 12.

Line 10070 determines the position, M, of the left-hand safe castle, 32809 is the screen map position of the top left-hand corner of our frame.

Line 10080 assigns the position and character code for the left-hand safe castle to array elements G(0) and G(1) respectively. L and MP are temporary position variables, M is used throughout as you position *now* and W is your position when you next move (the look-ahead variable). All are set to position of the safe castle.

We now have to work out the course of the path; for the PET we use the two diagonal lines, POKE codes 77 and 78 (suggested symbols for other micros are given in Table 3). Using only these two symbols for the path enables us to make a fairly simple decision on how the path may be drawn:

1) If the path is already slanting up to the right then only two possibilities are acceptable; upwards to the North East or downwards to the East (remember the path must be continuous).

2) If the path is already slanting down to the right then only downwards to the South East or upwards to the East are permissible.

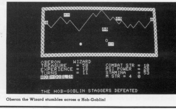
The choice of an upwards or downwards diagonal is made randomly in line 10100. Lines 10110 and 10120 initially set the POKE code, PC, for a downwards diagonal to be to the South East of the present path (L1 = L + 41) and for an upwards diagonal to be to the North East (L1 = L − 39). L1 is a temporary position variable.

In line 10130 we check to see if the path is trying to go through the top or bottom frames of the Valley. If it is, the program is directed back to line 10100 to choose again. If it is within the Valley, we allocate in line 10140 an element array G() to the POKE code for that section of the path. Looking back to 1) and 2) above, you will notice that if the next path element is different to the previous element it *must* always be to the East. This condition is checked in line 10150 and the temporary variable, L1, is altered if necessary.

Line 10160 stores the position of the path in array G(), assigns the starting position for the next section of the path to variable L and draws the section just computed on the screen.

This selection of path direction and position is repeated 36 times within a FOR . . . NEXT loop. Position 1 and 37 are safe castles from which you may leave the Valley if you so wish. Even elements of G() are screen positions and odd elements are POKE codes for the safe castles or the path. Line 10180 completes the 'path draw' by placing the the right-hand safe castle on the last path position.

Going back to line 10060 we see that this tests to see if the path has already been



```
OBERON    WIZARD          COMBAT STR = 10
TREASURE   = 58           PSI POWER  = 28
EXPERIENCE = 18           STAMINA    = 53
TURNS      = 11           M STR = 4   0
HOB-GOBLIN

THE HOB-GOBLIN STAGGERS DEFEATED
```

Oberon the Wizard stumbles across a Hob-Goblin!

computed. If it has not then G(0) will be zero. If it has, the program skips the path computation and jumps to line 10190 which initiates a FOR . . . NEXT loop that draws the path using the information stored in array G(). In fact if the path has been drawn for the first time (on entering the Valley) it is also redrawn in lines 10190 to 10210.

Having drawn the Valley path we now have to work out where the different scenarios are to be placed within the Valley. These positions are stored in array S(). Line 10220 checks if they have already been assigned and if they have, the program jumps to line 10280 where the scenarios are drawn on the map of the Valley.

Line 10240 generates two random numbers that represent row and column positions within the Valley. Line 10250 assigns this position to an element of array S(). Line 10260 checks to see that the chosen position and the position immediately to its right are not the path or safe castle (ie an empty space POKE code 32). If the positions chosen are free then the selection is repeated for the next element of array S().

Lines 10280-10300 POKE the scenario symbols on the screen. There is a chance that some symbols may be overwritten by other scenarios but then this is the luck of the draw! Woods are drawn first (code 216) then Swamps (code 173) and finally the Black Tower of Zaexon (pronounced Zeeks-on), code 87. Woods and Swamps are represented by two symbols side by side and both scenarios are repeated. The Black Tower is a single symbol; only one such Tower is found in the Valley.

Line 10310 assigns your present position to that of temporary variable, MP. This is used to remember your last position in the Valley when you return to the Valley from one of the other scenarios.

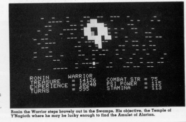
```
 9999  REM ** SCENARIO 1 (THE VALLEY)
10000  PRINT "[CLS]":F$="VAEGH":FL=1:S=1
10009  REM ** DRAW THE VALLEY FRAME
10010  PRINT "[HOM][REV][39"V][OFF]"
10020  FOR I=1 TO 12
10030  PRINT "[REV]["V][OFF][37 SPC][REV]["V][OFF]"
10040  NEXT I
10050  PRINT "[REV][39"V][OFF]"
10059  REM ** IF PATH ALREADY DRAWN SKIP
10060  IF G(0)<>0 THEN 10190
10069  REM ** COMPUTE THE PATH
10070  M=32809+(INT(RND(TI)*11+1)*40)
10080  L=M:MP=M:W=M:G(0)=M:G(1)=219
10090  FOR I=2 TO 72 STEP 2
10100  IF RND(TI)>0.5 THEN 10120
10110  PC=77:L1=L+41:GOTO 10130
10120  PC=78:L1=L-39
10130  IF L1>=33286 OR L1<=32806 THEN 10100
10140  G(I+1)=PC
10150  IF I>2 AND G(I+1)<>G(I-1) THEN L1=L+1
10160  G(I)=L1:L=L1:POKE G(I),G(I+1)
10170  NEXT I
10180  G(73)=219
10189  REM ** PLOT IN PATH
10190  FOR I=0 TO 72 STEP 2
10200  POKE G(I),G(I+1)
10210  NEXT I
10220  IF S(0)<>0 THEN 10280
10229  REM ** COMPUTE SCENARIO POSITIONS
10230  FOR I=0 TO 4
10240  N1=INT(RND(TI)*11)+1:N2=INT(RND(TI)*34)+1
10250  S(I)=32809+(40*N1)+N2
10260  IF PEEK(S(I))<>32 OR PEEK(S(I)+1)<>32 THEN 10240
10270  NEXT I
10279  REM ** PLOT IN SCENARIOS
10280  POKE S(0),216:POKE S(0)+1,216:POKE S(1),216:POKE
       S(1)+1,216
10290  POKE S(2),173:POKE S(2)+1,173:POKE S(3),173:POKE
       S(3)+1,173
10300  POKE S(4),87
10310  M=MP:W=M  .
10320  RETURN
```

WOODS AND SWAMPS

These two scenarios are fundamentally the same, only the characters representing their contents and the special scene in the middle of the Lake differ. The starting point selected by the Scene Control routine is 12000 for the Swamp and 12010 for the Woods. The function of each of these entry points is to establish the valid monster string, F$, and to assign the POKE code, PC, to the correct value for the scene. The routine proper begins by re-assigning the POKE value of the square under your feet to 32, a space, so that when you move away you don't leave the character you were standing on in the Valley behind you.

The screen is now cleared in preparation for the construction of the scene. A pointer variable, L, is set to the value of the top left-hand corner of the scene and a random FOR . . . NEXT loop inserts 200 appropriate scene characters into the screen area. This is all handled by the block of code from 12040 to 12070.

Having now constructed the inside of the scene, we have to print a border around it in order to detect an attempt at movement outside the scene



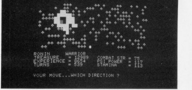
```
RONIN         WARRIOR
TREASURE    = 14126     COMBAT STR = 75
EXPERIENCE  = 10240     PSI POWER  = 115
TURNS       = 595       STAMINA    = 113
```

Ronin the Warrior steps bravely out in the Swamps. His objective, the Temple of Y'Nagioth where he may be lucky enough to find the Amulet of Alarian.

area. The border is printed by the section of program between 12150 and 12190 and is made up of non-32 type spaces on the PET; other suggestions are to be found in Table 3. Because the border is printed in after the scene has been constructed, it will overwrite any stray scene characters in the 1st and 40th column; this simplifies the random printing routine. The semi-colons at the end of lines 12150 and 12170 ensure that the frame is continuous; printing into the 40th column would otherwise force a Carriage Return.

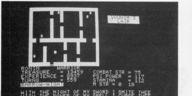
The next operation is to POKE a space character to the position in which the character will appear and assign W to the address of that position. This ensures that when you enter the Woods you are not completely hemmed in by trees; it doesn't matter in the case of the Swamps.

Because you could re-enter the Woods and Swamps from one of the secondary scenes, a check is made in line 12210 to see if the POKE code of the square you were standing on was a doorway. If it was, your current position is reset to the position you were in when you entered that secondary scene. The value of that position is then held in the variable MW and is assigned when you enter a secondary scene.



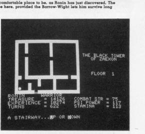
```
RONIN      WARRIOR
TREASURE   = 12989   COMBAT STR = 75
EXPERIENCE = 8654    PSI POWER  = 117
TURNS      = 539     STAMINA    = 113

YOUR MOVE...WHICH DIRECTION ?
```

The alternative secondary scenario to the Swamps is the Woods wherein can be found the dread Vounim's Lair.



```
                                   VOUNIM'S
                                     LAIR
RONIN      WARRIOR
TREASURE   = 13459   COMBAT STR = 75
EXPERIENCE = 8937    PSI POWER  = 117
TURNS      = 559     STAMINA    = 112
BARROW-WIGHT         M STR = 0  18

WITH THE MIGHT OF MY SWORD I SMITE THEE
WITH THE POWER OF MY SPELL I CURSE THEE
BURN YE SPAWN OF HELL AND SUFFER...
```

Vounim's Lair is an uncomfortable place to be, as Ronin has just discovered. The Helm of Evanna may be here, provided the Barrow-Wight lets him survive long enough to find it!



```
                           THE BLACK TOWER
                              OF ZAEXON

                              FLOOR  1

RONIN      WARRIOR
TREASURE   = 14126   COMBAT STR = 75
EXPERIENCE = 10274   PSI POWER  = 117
TURNS      = 622     STAMINA    = 113

A STAIRWAY...UP OR DOWN
```

The final alternative scenario is the Black Tower of Zaexon where the Amulet stones are hidden. The higher you go the more deadly the monsters.

```
11999  REM ** SCENARIO 2 (WOODS AND SWAMPS)
12000  F$="AFL":PC=45:GOTO 12020
12010  F$="FAEHL":PC=88
12020  PK=32
12030  PRINT "[CLS]"
12039  REM ** DRAW RANDOM WOODS OR SWAMPS
12040  L=32810
12050  FOR I=1 TO 200
12060  POKE L+INT(RND(TI)*515),PC
12070  NEXT I
12079  REM ** PRINT IN LAKE
12080  PRINT "[HOM]";D2$;R2$;"[2 CR][REV][2^SPC][OFF]"
12090  PRINT R2$;"[CR][REV][5^SPC][OFF]
12100  PRINT R2$;"[REV][2^SPC][OFF][2 SPC][REV][2^SPC]
       [OFF]"
12110  PRINT R2$;"[REV][2^SPC][^&][OFF][SPC][REV][3^SPC]
       [OFF]"
12120  PRINT R2$;"[CR][REV][4^SPC][OFF][CR][REV][2^SPC]
       [OFF]"
12130  PRINT R2$;"[3 CR][REV][2^SPC][OFF]"
12140  PRINT R2$;"[4 CR][REV][^SPC][OFF]"
12149  REM ** DRAW IN THE FRAME
12150  PRINT "[HOM][40^SPC]";
12160  FOR I=1 TO 13
12170  PRINT "[^SPC][38 CR][^SPC]";
12180  NEXT I
12190  PRINT "[40^SPC]"
12200  POKE 33306,32:W=33306
12210  IF Q1=104 THEN M=MW:W=M
12220  RETURN
```

THE BLACK TOWER

Our other primary scenario is the Black Tower of Zaexon. This is a six floor castle-type scene and its construction is also used to produce the secondary single floor scenes found in the Woods and Swamps. The Tower has a stable floor pattern; once a floor has been entered it will remain the same as long as you are in the Tower.

Scenario Control directs the program to the routine at 14000 assigning the monster string, F$; zeroing the floor pattern variable, P; and setting the room depth variable, H, to the current FLth element of array N(). The current position character is set to a space and the program jumps to 14020. The variables for the secondary scenes are initialised in 14010. One slight change is that the array P() is set to the value of P(2). This is done because the secondary scenes have initial FL values of 6 or 7 depending on type and these elements of P() are 0 which would cause the room pattern to be the same each time (see line 14070).

The frame of the Tower is printed first by lines 14020 to 14060 using a reversed space character. The vertical walls are drawn next by the somewhat complex routine found between 14070 and 14250. In order to ensure that the pattern of rooms varies on each floor and on each visit to the scene, we use the 31 element DATA statement at line 60000. These are READ sequentially for each new floor and represent the width of each room. To give variety to the pattern of rooms the starting point of the READ is determined randomly in the Scenario Control section and stored in P(2). To start the drawing sequence the DATA pointer is RESTOREd and then P(2) dummy READs are made; V is used only as a temporary store. Once again we use the pointer variable, L, to hold the address of the top left-hand corner of the scene. We now read the next three DATA items from the list and store them in array D(); the number of sets of 3 is stored in the temporary variable, P.

The actual drawing of the vertical walls is done by lines 14150 to 14240 and their length is dependent on the value of H, the room depth variable. The wall characters are POKEd into position as are the doors which occur a predetermined distance along them. Having drawn the first set of vertical walls the starting point is re-assigned in line 14250 and the next set is drawn in — this process is repeated until the walls have reached the bottom of the frame.

The horizontal walls can now be drawn in and their spacing is dependent on the value of the current element of array N(). The routine is located between lines 14270 and 14340.

As only the Black Tower has stairs, line 14350 causes the secondary scenes to skip over this section of the program. The Black Tower has stairs located in opposite corners for each floor and these are POKEd into position on lines 14360 and 14370. If you are on the ground floor of the Tower or in one of the secondary scenes, a doorway is POKEd into position by line 14380.

If you are stepping into the Tower or either of the secondary scenes for the first time, your character will be placed just inside the doorway; the check for this made in 14390 as P(3) will only be 0 if you haven't gone up any stairs yet.

The appropriate name for the castle-type scene is PRINTed into position by lines 14400 to 14480 and, in the case of the Tower, the floor number is also displayed.

Treasure can be found in the upper floors of the Tower and either of the two secondary scenes, provided the value of FL is equal to or greater then 4 *and* a random factor is greater than 0.3. If these conditions are not met, control returns to the Scenario Control section and then back to the Movement routine. If both conditions are met, a random number of special treasure symbols are displayed; between 2 and 6 can appear and are shown as asterisks. They are positioned by the two temporary variables N1 and N2 which act as row and column co-ordinates. Provided the position selected is vacant an asterisk is POKEd into place.

```
13999 REM ** SCENARIO 3 (CASTLE-TYPES)
14000 F$="CAGE":P=0:H=N(FL):PK=32:GOTO 14020
14010 F$="CBE":P=0:H=N(FL):PK=32:P(FL)=P(2)
14019 REM ** DRAW FRAME
14020 PRINT "[CLS][REV][2 CR][21 SPC][OFF]"
14030 FOR I=1 TO 13
14040 PRINT "[REV][2 CR][SPC][OFF][19 SPC][REV][SPC]
      [OFF]"
14050 NEXT I
14060 PRINT "[REV][2 CR][21 SPC][OFF]"
14069 REM ** DRAW VERTICAL WALLS
14070 RESTORE:FOR I=1 TO P(FL)
14080 READ V:IF V=100 THEN RESTORE
14090 NEXT I
14100 L1=32810
14110 FOR J=1 TO 3
14120 READ D(J):P=P+1
14130 IF D(J)=100 THEN RESTORE:D(J)=3:P=P+1
14140 NEXT J
14150 FOR I=0 TO H:PC=160
14160 L=L1+(40*I):IF L>33290 THEN 14260
14170 IF I=1 THEN PC=32
14180 IF D(1)=0 THEN PC=160:GOTO 14200
14190 POKE L+D(1),PC:PC=160
14200 IF I=3 THEN PC=32
```

```
14210 POKE L+D(1)+D(2),PC:PC=160
14220 IF I=4 THEN PC=32
14230 POKE L+D(1)+D(2)+D(3),PC:PC=160
14240 NEXT7 I
14250 L1=L1+(40*H)+40:GOTO 14110
14259 REM ** DRAW HORIZONTAL WALLS
14260 L1=32810
14270 FOR J=1 TO 4
14280 L=L1+(40*J*(H+1))
14290 FOR K=1 TO 19
14300 IF L>33250 THEN 14350
14310 POKE L+K,PC
14320 IF K=2 OR K=3*H OR K=17 THEN POKE L+K,32:
      POKE L+K-40,32:POKE L+K+40,32
14330 NEXT K
14340 NEXT J
14349 REM ** DRAW IN THE STAIRS
14350 IF S=5 OR S=6 THEN 14380
14360 IF FL/2=INT(FL/2) THEN POKE 33291,102:GOTO 14380
14370 POKE 32829,102
14380 IF FL=2 OR S=5 OR S=6 THEN POKE 33336,104:
      POKE 33296,32
14390 IF P(3)=0 THEN W=33296
14399 REM ** WRITE APPROPRIATE NAME
```

```
14400  IF S=5 THEN 14470
14410  IF S=6 THEN 14450
14420  PRINT "[HOM]";R1$;"[4 CD][3 CR]THE BLACK TOWER"
14430  PRINT R1$;"[3 CR][3 SPC]OF ZAEXON"
14440  PRINT R1$;"[3 CD][3 CR][3 SPC]FLOOR ";FL-1:
       GOTO 14490
14450  PRINT "[HOM]";R1$;"[2 CD][5 CR][REV] VOUNIM'S
       [SPC][OFF]"
14460  PRINT R1$;"[5 CR][REV][3 SPC]LAIR[3 SPC][OFF]":
       GOTO 14500
14470  PRINT "[HOM]";R1$;"[2 CD][4 CR][REV]THE TEMPLE OF
       [OFF]"
14480  PRINT R1$;"[4 CR][REV][2 SPC]Y'NAGIOTH[2 SPC]
       [OFF]"
14490  P(FL+1)=P(FL)+P
14499  REM ** SCATTER SPECIAL FINDS
14500  IF FL<4 OR RND(TI)<0.3 THEN RETURN
14510  FOR I=1 TO INT(RND(TI)*5)+2
14520  N1=INT(RND(TI)*19)
14530  N2=INT(RND(TI)*12)
14540  IF PEEK(32811+40*N2+N1)<>32 THEN 14520
14550  POKE (32811+40*N2+N1),42
14560  NEXT I
14570  RETURN
```

STAIRS

In the Black Tower each floor is connected to the next by a set of stairs. These are set at diagonally opposite corners of each floor and each stair operates only in one direction. This means that if you walk up one flight you have to cross the entire floor to reach the next set; you can't simply go down the flight you came up!

The routine is located from 15000 to 15100 and starts by offering you a choice of going either up or down. The character pressed is checked at 15030 to 15050 to see if it is valid and if it is, the FL variable is incremented. This value represents the floor to which you wish to move and checked by line 15060 to ensure that it is within limits. If the value of FL is outside the limits, a suitable message is printed and the current floor level reset into FL from the temporary variable TV.

```
14999 REM ** STAIRS ROUTINE
15000 POKE W,81:POKE M,32
15010 PRINT D$;"A STAIRWAY...
      UP OR DOWN ?":TV=FL
15020 VG$="UD":GOSUB 1500:REM
       ** UNIGET
15030 IF GC$="U" THEN FL=FL+1:
      GOTO 15050
15040 FL=FL-1
15050 IF FL>7 OR FL<2 THEN 15080
15060 DF=110:DL$="D":GOSUB
      36000:REM ** DELAY
15070 GOTO 9220
15080 PRINT D$;"THESE STAIRS
      ARE BLOCKED[SPC]"
15090 DF=60:DL$="D":GOSUB
      36000:REM ** DELAY
15100 FL=TV:GOTO 15010
```

DELAYS

The routine from 36000 can be broken down into three functional blocks; delay, wipe and update. All calls to the routine are first set up by defining the contents of the variable DF which controls the length of the delay. If only the delay section of the routine is required then a flag variable, DL$, is set to "D" to indicate this; the test in line 36020 causes an early RETURN.

In cases where a message wipe is needed after the delay but no update is required, the flag is set to "W" which forces a RETURN at line 36060. The wipe is simply performed by overwriting the text area with spaces.

The rest of the routine is concerned with updating the adventurer's status on the screen. Before the data is printed it is checked to see if it has reached or exceeded the maximum for the current character type; see Table 2. The code that performs these checks can be found in lines 36070 to 36100. The variables for experience, treasure and turns, can only increase so these are simply overprinted in lines 36120 to 36140. The value of combat strength, psi power and stamina can decrease as well as increase so these are first erased and then reprinted; lines 36150 to 36170 perform this task.

If a combat is in progress the flag variable, CF, is set to 1 and this is tested for in 36180. If it is set, the monster's current status is also updated at line 36210 and 36220. If, however, the flag is cleared to show that no combat is taking place, the line of the screen where this information would normally occur is wiped clean.

```
35999 REM ** DELAY, WIPE & UPDATE ROUTINE
36000 FOR DL=1 TO (DF*TM)
36010 NEXT DL
36020 IF DL$="D" THEN DL$="":RETURN
36030 PRINT D$;SP$
36040 PRINT SP$
36050 PRINT SP$
36060 IF DL$="W" THEN DL$="":RETURN
36070 IF CS>77-INT(2*P1^2.5) THEN CS=77-INT(2*
      P1^2.5)
36080 IF PS<7 THEN PS=7
36090 IF PS>INT(42*(P1+1)^LOG(P1^3.7))+75 THEN
      PS=INT(42*(P1+1)^LOG(P1^3.7))+75
36100 IF C>125-(INT(P1)*12.5) THEN C=125-INT(INT(P1)*
      12.5)
36110 PRINT D1$;"[CU]";J$,P$
36120 PRINT "TREASURE   =";TS
36130 PRINT "EXPERIENCE =";EX
36140 PRINT "TURNS      =";TN
36150 PRINT D1$;R1$;"COMBAT STR =[4 SPC][4 CL]";CS
36160 PRINT R1$;"PSI POWER  =[4 SPC][4 CL]";PS
36170 PRINT R1$;"STAMINA    =[4 SPC][4 CL]";C
36180 IF CF=1 THEN 36210
36190 PRINT SP$
36200 RETURN
36210 PRINT D$;[2 CU][REV]";M$;"[OFF]";
36220 PRINT D$;R1$;"[2 CU]M STR =[12 SPC][12 CL]";MS;N;
      "[4 SPC]"
36230 RETURN
```

RATINGS

The ratings system used in the Valley program is based on a character achieving the maximum rating of 28, Master of Destiny, only after amassing 200,000 experience points.

Assigning a rating of 7 to an experience of 10,000 and a rating of 20 to 50,000 experience, the plotted curve began to show definite parabolic tendencies. After experimenting with the general equation of a parabola, $y^2 = 4ax$ or $y = c\sqrt{x}$ (where c is a constant), no simple values were found to fit. So . . we compromised! Using the formula, $y = 0.067\sqrt{x}$, we managed to get y values of 6.7 for an x value of 10,000, 15 at 50,000 and 28 at 200,000.

Realising that the rating should be based on experience and treasure, the x factor was defined as $x = EX + TS/3$. Then, in an attempt to penalise cowardice and rewarding those taking risks, a second factor, $\log(EX/(TN + 1) \uparrow 1.5))$, was added taking the number of turns to acquire your experience into the final equation.

1	Monster Food	15	Necromancer
2	Peasant	16	Loremaster
3	Cadet	17	Paladin
4	Cannon Fodder	18	Superhero
5	Path Walker	19	Dragon Slayer
6	Novice Adventurer	20	Knight of the Valley
7	Survivor	21	Master of Combat
8	Adventurer	22	Dominator
9	Assassin	23	Prince of the Valley
10	Apprentice Hero	24	Guardian
11	Giant Killer	25	War Lord
12	Hero	26	Demon Killer
13	Master of the Sword	27	Lord of the Valley
14	Champion	28	Master of Destiny

Above: The rating numbers developed during the game and the character classifications that they correspond to. They may be easily-added into versions of the Valley running on machines with more than 16K.

```
44999  REM ** RATING ROUTINE
45000  DF=5:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
45010  RT=INT(0.067*(EX+TS/3)^0.5+LOG(EX/((TN+1)^1.5))):
       IF RT>28 THEN RT=28
45020  IF RT<0 THEN RT=0
45030  PRINT D$;"YOUR RATING NOW BE";RT
45040  IF T(2)=1 THEN PRINT "YOU HAVE THE HELM OF EVANNA"
45050  IF T(0)=1 THEN PRINT "AMULET STONES...[SPC]";T(1)
45060  DF=250:DLS="W":GOSUB 36000:REM ** DELAY + WIPE
45070  IF GC$="E" THEN C=C-10:GC$="":GOTO 2010:
       REM ** MOVEMENT
45080  RETURN
```

QUIT

If the adventurer steps on either of the two safe castles, one at each end of the path, he is offered the option of leaving the Valley. Regardless of his selection, his current rating is also computed and displayed at this point. If the player chooses to leave the Valley by keying "Y", control is passed to the Save routine at line 50000.

Because the castle is safe the player's character is 'healed' of his wounds and readied for the Valley once more. This healing consists of resetting the stamina to its maximum value and ensuring a minimum combat strength of 20.

```
47999  REM ** QUIT VALLEY ROUTINE
48000  PRINT D$;"THOU ART SAFE IN A CASTLE":IF CS<20 THEN
       CS=20
48010  POKE M,PK:PK=PEEK(W):M=W:POKE M,Q
48020  PRINT "WILT THOU LEAVE THE VALLEY (Y/N) ?"
48030  VG$="YN":GOSUB 1500:REM ** UNIGET
48040  DF=5:DLS="W":GOSUB 36000:REM ** DELAY + WIPE
48049  REM ** GENERATE RATING IN CASE OF SAVE
48050  GOSUB 45000:REM ** RATING
48060  DF=110:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
48070  IF GC$="Y" THEN 50000:REM ** SAVE ROUTINE
48080  C=150:PRINT D$;"THY WOUNDS HEALED...THY SWORD
       SHARP"
48090  PRINT "GO AS THE GODS DEMAND..TRUST NONE OTHER"
48100  DF=120:GOSUB 36000:REM ** DELAY + UPDATE
48110  GOTO 2010:REM ** MOVEMENT
```

SAVE

Stepping on one of the two safe castles is the only way to leave the Valley in an upright position as the option to save your character on tape is then offered. Taking this option out of the Quit routine passes control to the Save routine at 50000.

```
49999  REM ** SAVE CHARACTER ROUTINE
50000  PRINT "[CLS]DO YOU WISH TO SAVE ";J$;" ?"
50010  PRINT:PRINT "PLEASE KEY Y OR N"
50020  VGS="YN":GOSUB 1500:REM ** UNIGET
50030  IF GCS="N" THEN 50210
50040  PRINT "[CLS]PLACE YOUR CASSETTE IN THE TAPE DECK"
50050  PRINT "IS IT REWOUND ?"
50060  GOSUB 1600:REM ** ANYKEY
50069  REM ** THIS IS FOR PET ONLY
50070  OPEN 1,1,1,J$
50080  PRINT#1,P$
50090  PRINT#1,TS
```

```
50100  PRINT#1,EX
50110  PRINT#1,TN
50120  PRINT#1,CS
50130  PRINT#1,PS
50140  PRINT#1,T(0)
50150  PRINT#1,T(1)
50160  PRINT#1,T(2)
50170  PRINT#1,C1
50180  PRINT#1,P1
50190  CLOSE 1
50200  PRINT "[CLS][3 CD]"," *** DONE ***"
50210  PRINT D$;"[6 SPC]TYPE RUN TO START AGAIN"
50220  CLR
50230  END
```

At the end of this routine, whether you reach it by saving the data on tape or by choosing not to save in the Quit routine and dropping through, all the current variables are cleared and a farewell message displayed.

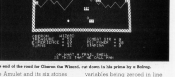
DEATH

This routine is the one part of the program the player would rather not have executed! Many and varied are the ways in which one can arrive at line 55000 and on all but one occasion the outcome is inevitable. The one exception is when you have been fortunate enough to collect the Amulet of Alarian and filled it with the six missing stones because this gives you a second life.

The test to see if you have the Amulet and its stones is made at 55020 and if successful you are restored to life. The price is, however, high as you lose all your treasure together with the Amulet and its stones. Your combat strength and your psi power are both set to 30; the only value that remains the same after death is your experience.

If, as is most likely the case, you don't have the protection of the Amulet and its six stones then the game ends with all the variables being zeroed in line 50220.

```
54999  REM ** DEATH ROUTINE
55000  C=0:CS=0:PS=0:CF=0
55010  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
55020  IF T(1)=6 THEN 55070
55030  PRINT D$,"[CR]OH WHAT A FRAIL SHELL"
55040  PRINT,"[2 CR]IS THIS THAT WE CALL MAN"
55050  DF=300:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
55060  PRINT "[CLS]":GOTO 50210
55069  REM ** RESTORE CHARACTER TO LIFE
55070  T(0)=0:T(1)=0:TS=0:CS=30:C=150:PS=30
55080  PRINT D$;"ALARIAN'S AMULET PROTECTS THY SOUL"
55090  PRINT "[CD][REV][2 SPC]LIVE AGAIN[2 SPC][OFF]"
55100  DF=150:GOSUB 36000:REM ** DELAY + UPDATE
55110  L=G(0):MP=L:M=W:S=1:GOTO 9220:REM ** SCENE CONTROL
```



```
OBERON     WIZARD
TREASURE    = 58       COMBAT STR = 0
EXPERIENCE  = 20       PSI POWER  = 28
TURNS       = 12       STAMINA    = 0

          OH WHAT A FRAIL SHELL
         IS THIS THAT WE CALL MAN
```

The end of the road for Oberon the Wizard, cut down in his prime by a Balrog.

DATA

Rather than placing each data block with its relevant routine we have chosen to lump it all together at the end of the program. The first block contains all the information needed to build the three castle-type scenarios (see the relevant sections for more details on this). The second block of data holds the monster information which is READ into the three arrays M$(), MSO and N1() at the start of the progam.

```
59999  REM ** DATA FOR CASTLE TYPE SCENARIOS
60000  DATA 4,7,3,6,4,4,6,5,3,6,0,3,8,4,3,5,5,3,
       8,3,4,5,0,6,3,6,4,6,4,7,4,100
60009  REM ** DATA FOR MONSTERS
60010  DATA AWOLFEN,9,0,AHOB-GOBLIN,9,0,AORC,9,0,
       EFIRE-IMP,7,3,GROCK-TROLL,19,0
60020  DATA EHARPY,10,12,AOGRE,23,0,BBARROW-WIGHT,0,25,
       HCENTAUR,18,14
60030  DATA EFIRE-GIANT,26,20,VTHUNDER-LIZARD,50,0,
       CMINOTAUR,35,25,CWRAITH,0,30
60040  DATA FWYVERN,36,12,BDRAGON,50,20,
       CRING-WRAITH,0,45,ABALROG,50,50
60049  REM ** SPECIAL MONSTERS FOR WATER ONLY
60050  DATA LWATER-IMP,15,15,LKRAKEN,50,0
```

CREDITS

The Valley was devised and programmed by Peter Freeby, Peter Green, Ron Harris and Henry Budgett during 1981. The story of The Valley was written by Roger Munford.

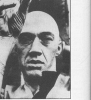
Those wishing to avoid typing in the 16K of code or wishing to adapt the program for other Commodore machines will be interested to know that versions are available from ASP Software to run on CBM 2000/3000/4000, VIC-20, Commodore 64 and 8000 series systems. Tapes are priced at £11.45 all inclusive and the 8000 series version is available on an 8050 format disc for £13.95 inclusive. Orders should be sent to ASP Software at 145 Charing Cross Road, London WC2H 0EE.

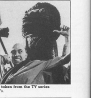
TOWERS OF BRAHMA

The object of the game is to transfer rings from one pillar to another, although it's not as easy as you might think.

```
100 PRINT CHR$(147);"THE TOWER OF BRAHMA"
110 DIM A(3,9)
120 PRINT
130 PRINT "LOOKS LIKE THIS (WITH 5 RINGS):-"
140 PRINT
150 PRINT "          A          B        C"
160 PRINT "         1A1         B        C"
170 PRINT "        22A22        B        C"
180 PRINT "       333A333       B        C"
190 PRINT "      4444A4444      B        C"
200 PRINT "     55555A55555     B        C"
210 PRINT "***********************************"
220 PRINT
230 PRINT "THE OBJECT OF THE GAME IS TO TRANSFER"
240 PRINT "THE RINGS FROM PILLAR 'A' TO PILLAR 'C',"
250 PRINT "ONE AT A TIME."
260 PRINT
270 PRINT "AT NO TIME CAN A LARGER RING BE ON TOP"
280 PRINT "A SMALLER RING."
290 PRINT
300 PRINT "YOU MAY REQUEST A PICTURE AT ANY TIME BY"
310 PRINT "TYPING 'P' AS THE NEXT 'FROM' COMMAND."
320 PRINT
330 PRINT
340 INPUT "HOW MANY RINGS WOULD YOU LIKE (5-9)[9 SPC][9 CL][2 CR]*[3 CL]";Z$
350 IF Z$="*" THEN PRINT "[2 CU]":GOTO 340
360 Z=VAL(Z$)
370 IF Z<5 THEN PRINT "[2 CU]":GOTO 340
380 IF Z>9 THEN PRINT "[2 CU]":GOTO 340
390 PRINT CHR$(147)
400 TI$="000000"
410 FOR X=1 TO 3
420 FOR Y=1 TO Z
430 IF X=1 THEN A(X,Y)=Y
440 IF X>1 THEN A(X,Y)=0
450 NEXT Y
460 NEXT X
470 INPUT "TAKE A RING FROM [2 CR]*[3 CL]";J$
480 Q=Q+1
490 J$=LEFT$(J$,1)
500 IF J$="*" THEN PRINT "[ 2CU]":GOTO 470
510 IF J$="P" THEN PRINT CHR$(147):GOTO 760
520 IF J$="S" THEN 990
530 IF J$<"A" THEN 470
540 IF J$>"C" THEN 470
550 V=ASC(J$)-64
560 FOR Y=1 TO Z
570 IF A(V,Y)>0 THEN T=A(V,Y):A(V,Y)=0:GOTO 610
580 NEXT Y
590 PRINT "NO RING ON THAT PILLAR"
600 GOTO 470
610 INPUT "TRANSFER TO [2 CR]*[3 CL]";K$
620 K$=LEFT$(K$,1)
630 IF K$="*" THEN PRINT "[2 CU]":GOTO 610
640 IF K$="S" THEN 990
650 IF K$<"A" THEN 610
660 IF K$>"C" THEN 610
670 U=ASC(K$)-64
680 FOR W=Z TO 1 STEP-1
690 IF A(U,W)=0 THEN IF W=Z THEN A(U,W)=T:GOTO 470
700 IF A(U,W)=0 THEN IF W<Z THEN IF A(U,W+1)<T THEN 730
710 IF A(U,W)=0 THEN A(U,W)=T:GOTO 840
720 NEXT W
730 PRINT "THE RING ON THAT PILLAR IS SMALLER"
740 A(V,Y)=T
750 GOTO 470
760 FOR Y=1 TO Z
770 FOR X=1 TO 3
780 PRINT TAB(5*X);A(X,Y);
790 NEXT X
800 PRINT
810 NEXT Y
820 PRINT
830 GOTO 470
840 FOR Y=1 TO Z
850 IF A(3,Y)<>Y THEN 470
860 NEXT Y
870 PRINT CHR$(147)
880 PRINT "CONGRATULATIONS, YOU'VE DONE IT AND IT ONLY"
890 PRINT "TOOK YOU ";
900 PRINT MID$(TI$,3,2);" MINS ";RIGHT$(TI$,2);" SECS"
910 PRINT "AND ";Q;" MOVES"
920 PRINT
930 INPUT "TRY AGAIN [2 CR]*[3 CL]";A$
940 IF LEFT$(A$,1)="*" THEN PRINT "[2 CU]":GOTO 930
950 IF LEFT$(A$,1)="Y" THEN 330
960 IF LEFT$(A$,1)<>"N" THEN 930
970 PRINT "OK. GOODBYE"
980 END
990 PRINT "PITY, SO FAR IT";
1000 GOTO 890
```

The following game is a simulation of the (supposed) project given by the supreme Hindu deity, Brahma, to his disciples. The object of the game is explained in the initial text and is not as simple as you might think from first inspection.





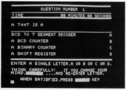
Photos taken from the TV series *Kung Fu*.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

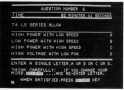
EDUCATIONAL

Micro Examination — Test your friends and children with this multiple choice program. Originally published in **Computing Today**, July 1980.

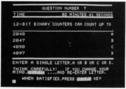
Quiz Time — Assess your performance in terms of speed and accuracy with a multiple choice program.
Originally published in **Computing Today**, September 1982.



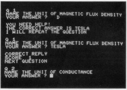
```
             QUESTION NUMBER  1
TIME                 00 MINUTES 00 SECONDS

A 7447 IS A

BCD TO 7 SEGMENT DECODER                 A
A BCD COUNTER                            B
A BINARY COUNTER                         C
A SHIFT REGISTER                         D

ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,DELETE ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS RETURN KEY
■
```



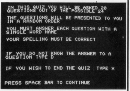
```
             QUESTION NUMBER  6
TIME                 02 MINUTES 11 SECONDS

74 LS SERIES ALLOW

HIGH POWER WITH LOW SPEED                A
LOW POWER WITH HIGH SPEED                B
HIGH POWER WITH HIGH SPEED               C
HIGH VOLTAGE WITH LOW POW                D

ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,DELETE ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS RETURN KEY
■
```



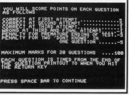
```
             QUESTION NUMBER  7
TIME                 02 MINUTES 41 SECONDS

12-BIT BINARY COUNTERS CAN COUNT UP TO

2048                                     A
2047                                     B
4098                                     C
4097                                     D

ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,DELETE ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS RETURN KEY
■
```



```
Q 1
NAME THE UNIT OF MAGNETIC FLUX DENSITY
YOUR ANSWER ?   D

YOU NEED HELP!
THE CORRECT ANSWER IS TESLA
I WILL REPEAT THE QUESTION

Q 1
NAME THE UNIT OF MAGNETIC FLUX DENSITY
YOUR ANSWER ? TESLA

CORRECT REPLY
SCORE  1
NEXT QUESTION

Q 2
NAME THE UNIT OF CONDUCTANCE
YOUR ANSWER ? ■
```



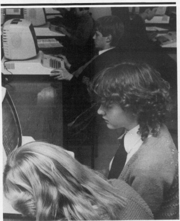
```
IN THIS QUIZ YOU WILL BE ASKED 20
QUESTIONS OUT OF THE POSSIBLE 24

THE QUESTIONS WILL BE PRESENTED TO YOU
IN A RANDOM ORDER

YOU MUST ANSWER EACH QUESTION WITH A
SINGLE WORD NAME

YOUR SPELLING MUST BE CORRECT


IF YOU DO NOT KNOW THE ANSWER TO A
QUESTION TYPE D

IF YOU WISH TO END THE QUIZ  TYPE X


PRESS SPACE BAR TO CONTINUE
```



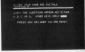
```
YOU WILL SCORE POINTS ON EACH QUESTION
AS FOLLOWS

CORRECT AT FIRST ATTEMPT ............5
CORRECT AT SECOND ATTEMPT ...........3
CORRECT AT THIRD ATTEMPT.............1
WRONG AT THIRD AND FINAL ATTEMPT....-1
PENALTY FOR PREMATURE ENDING OF TEST.-3
PENALTY FOR TWO 'DONT KNOWS' IN
                     ONE QUESTION ......-5

MAXIMUM MARKS FOR 20 QUESTIONS  ....100

EACH QUESTION IS TIMED FROM THE END OF
THE QUESTION PRINTOUT TO WHEN YOU HIT
THE RETURN KEY


PRESS SPACE BAR TO CONTINUE
```

MICRO EXAMINATION

Multiple choice exams represent an ideal entry point to the classroom for computers.



The multiple-choice question paper first vented its spite on thousands of the luckless 'volunteers' who were trained during the last World War. The traditional essay type examination was too slow, favoured those who had the ability to disguise their ignorance with high-sounding jargon and, worst of all, the marking of the exam required some degree of professionalism. A gentleman by the name of Ballard is credited with the invention of presenting a question and four answers labelled A, B, C and D . . . only one of which is considered to be the 'right' one. All the trainee had to do was place a cross in the 'right' place. The technique was highly successful. A wide range of subjects could be covered in 100 question paper and could be marked by unskilled personnel in less than a minute by simple placing a prepared stencil over the paper. Although originally intended as a wartime expedient, the advantages were found to be so great that it has survived until the present day. The educational Establishment was naturally very critical, mumbling something like ". . . .training a bunch of parrots etc etc" but the seal of respectability was finally given when technical colleges, and even universities, succumbed to the temptation. The computer is ideally situated as a tool in this area of education because it demands a minimum of keyboard interaction from the examinee. A question is flashed on the screen, demanding that ONE particular key is pressed. Traditional keyboard questions and answers suffer from the infuriating habit of marking you wrong even if a trivial spelling error is made or perhaps even an extra space.



```
ENTER YOUR NAME AND INITIALS

WHEN THE QUESTIONS APPEAR,KEY EITHER
A,B,C OR D.  OTHER KEYS IMPLY [illegible]
   PRESS ANY KEY WHEN YOU ARE READY
```

GUIDING PRINCIPLES

Much of the criticism of multiple choice papers is due not to the method itself but the style of the questions. Too many of these questions are made up by small-minded individuals who often lack real knowledge of their subject and make up for it by composing, what they believe to be, 'clever' tricks which are guaranteed to fool the poor student. The rules are simple:

a) keep the question short and straightforward.

b) make sure that all four of the answers are *superficially* correct.

c) the correct answer expected should be that which is more universally true.

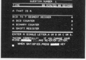
d) don't make one of the

answers absurdly wrong because this is equivalent of reducing the number of choices by one.

e) make sure you really know the correct answer yourself!

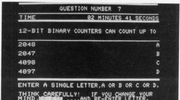
THE PROGRAMS

These have been in use for some time at a MOD Training Establishment (where I slave from dawn to dusk in return for the occasional bowl of rice). MULTIPLE CHOICE PREPARATION allows anyone to enter 25 questions, each with four answers and the right answer. In addition, the time allowed and the minimum pass mark can be entered. The end result of this activity is 'Data Tape' with the precious collection of mental sadism embeded within its magnetic bosom.

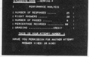


```
              QUESTION NUMBER  1
TIME                 00 MINUTES 00 SECONDS
A 7447 IS A
BCD TO 7 SEGMENT DECODER                 A
A BCD COUNTER                            B
A BINARY COUNTER                         C
A SHIFT REGISTER                         D
ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,[illegible] ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS [illegible] KEY
```

The second program, MULTIPLE CHOICE EXAM, is operated by the person being examined and begins with instructions for loading the data tape containing the questions. The questions are presented together with the four answers and the final score with percentages and a grading category appears on the screen. Should the examinee exceed the time allowed, the questions cease and the score page is presented immediately. Time lapse is given on each 'page'. Facilities exist during the preparation stage for producing the questions, displaying the questions, modifying them, saving them on the data tape and making additional copies from an existing tape. Subroutines are used to ensure that deficiencies in the data tape operating system (present in the 'old ROMS') are corrected by suitable patching. A subroutine for treating the keyboard as an INPUT FILE is also provided to prevent the program from breaking out should the operator inadvertently press RETURN before entering a character. The preamble for the questions are written as DATA/READ statements in order that modifications to suit local conditions are easy to incorporate. It is reasonably 'idiot proof' but to fit the program into an 8K PET the REM statements had to be curtailed. However, the program should be fairly straightforward to follow without them.



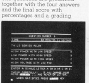
```
              QUESTION NUMBER  7
TIME                 02 MINUTES 41 SECONDS
12-BIT BINARY COUNTERS CAN COUNT UP TO
2048                                     A
2047                                     B
4098                                     C
4097                                     D
ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,DELETE ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS RETURN KEY
```



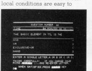
```
STUDENTS NAME ..GERRING P
|           PERFORMANCE ANALYSIS           |
| NUMBER OF RESPONSES ......         25   |
| RIGHT ANSWERS ......               20   |
| NUMBER OF PASSES ......             0   |
| PERCENTAGE RECORDED ............ 80     |
| GRADING ................CREDIT          |
     THIS IS YOUR ATTEMPT NUMBER  1
HAVE YOU PERMISSION FOR ANOTHER ATTEMPT
      ANSWER Y(YES) OR N(NO)
```

APPLICATION

Although the program is oriented towards the teaching profession, it could also prove useful in the home. It is educational in two senses; answering the questions which one member of the family has set with the aid of the 'PREPARATION' program and vice versa. It is probably harder to write a good set of 25 questions than it is to answer them. Some of the questions may of course be disputed (or rather the particular answer which is supposed to be correct) but even this is good. Plato and his followers spent most of their life learning by arguing. Because of the possibility of dispute, facilities are provided for modifying a question. Some ▶



```
              QUESTION NUMBER  6
TIME                 02 MINUTES 11 SECONDS
74 LS SERIES ALLOW
HIGH POWER WITH LOW SPEED                A
LOW POWER WITH HIGH SPEED                B
HIGH POWER WITH HIGH SPEED               C
HIGH VOLTAGE WITH LOW POW                D
ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,[illegible] ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS [illegible] KEY
```



```
              QUESTION NUMBER  16
TIME                 04 MINUTES 40 SECONDS
THE BASIC ELEMENT IN TTL IS THE
AND                                      A
OR                                       B
EXCLUSIVE-OR                             C
NAND                                     D
ENTER A SINGLE LETTER,A OR B OR C OR D.
THINK CAREFULLY!   IF YOU CHANGE YOUR
MIND,[illegible] ....AND RE-ENTER LETTER.
   WHEN SATISFIED,PRESS [illegible] KEY
```

The examination program not only records the answers given but also computes a final analysis on the student's performance.

```
100 DIM A$(25,7):L=25
110 GOTO 610
120 REM ** ENTER PREAMBLE TO ARRAY
130 PRINT CHR$(147)
140 F=0
150 PRINT "[39↑0]"
160 PRINT "ENTER TITLE OF EXAM"
170 GOSUB 1520
180 A$(0,0)=I$
190 PRINT:PRINT
200 PRINT "ENTER NAME OF PERSON COMPILING EXAM"
210 GOSUB 1520
220 A$(0,1)=I$
230 PRINT:PRINT
240 PRINT "ENTER DATE"
250 GOSUB 1520
260 A$(0,2)=I$
270 PRINT:PRINT
280 PRINT "ENTER MINIMUM PASS MARK EXPECTED"
290 GOSUB 1520
300 A$(0,3)=I$
310 PRINT:PRINT
320 PRINT "ENTER TIME ALLOWED (IN MINUTES)"
330 GOSUB 1520
340 A$(0,4)=I$
350 PRINT:PRINT
360 PRINT "[38↑0]"
370 PRINT:PRINT
380 GOSUB 1580
390 IF F=0 THEN 120
400 F=0
410 REM ** ENTER MAIN ARRAY
420 PRINT CHR$(147)
430 M=0
440 FOR N=1 TO L
450 FOR Q=1 TO 7
460 READ P$(Q)
470 IF Q=1 THEN PRINT P$(Q);N:A$(N,Q)=STR$(N):GOTO 530
480 PRINT P$(Q)
490 GOSUB 1520
500 PRINT
510 PRINT "[39↑#]"
520 A$(N,Q)=I$
530 NEXT
540 RESTORE
550 IF M=1 THEN RETURN
560 GOSUB 1580
570 IF F<>1 THEN PRINT CHR$(147):GOTO 450
580 F=0
590 PRINT CHR$(147)
600 NEXT
610 PRINT CHR$(147):PRINT TAB(240)
620 PRINT "[↑%]PREPARE 25 QUESTIONS ...............1[↑']"
630 PRINT
640 PRINT "[↑%]VIEW THE QUESTIONS .................2[↑']"
650 PRINT
660 PRINT "[↑%]MODIFY SELECTED QUESTION ..........3[↑']"
670 PRINT
680 PRINT "[↑%]SAVE QUESTIONS ON A TAPE ..........4[↑']"
690 PRINT
700 PRINT "[↑%]LOAD AN EXISTING QUESTION TAPE ....5[↑']"
710 PRINT:PRINT:PRINT
720 PRINT "      ENTER DESIRED OPTION NUMBER"
730 GET K$
740 IF K$="" THEN 730
750 IF VAL(K$)<1 OR VAL(K$)>5 THEN 730
760 ON VAL(K$) GOTO 130,1220,1120,770,1370
770 REM ** WRITE PREAMBLE TO TAPE
780 PRINT CHR$(147):PRINT TAB(120)
790 GOSUB 1650
800 PRINT "IT IS ASSUMED YOU HAVE A BLANK TAPE IN"
810 PRINT
820 PRINT "THE CASSETTE, IT IS REWOUND AND THE"
830 PRINT
840 PRINT "MOTOR IS SWITCHED OFF!"
850 PRINT:PRINT:PRINT
860 PRINT "BE PATIENT !"
870 PRINT:PRINT:PRINT
880 PRINT "OBEY INSTRUCTIONS."
890 PRINT:PRINT:PRINT
900 GOSUB 1470
910 PRINT CHR$(147):PRINT TAB(240)
920 GOSUB 1650
930 OPEN 1,1,2
940 FOR Q=0 TO 4
950 PRINT#1,A$(0,Q)
960 NEXT
970 REM ** WRITE TEXT COMPLETE
980 FOR N=1 TO L
990 FOR Q=1 TO 7
1000 PRINT#1,A$(N,Q)
1010 NEXT:NEXT
1020 GOSUB 1680
1030 CLOSE 1
1040 GOSUB 1470
1050 PRINT CHR$(147):PRINT TAB(240)
1060 PRINT "DATA NOW ON TAPE FILE"
1070 PRINT "[21↑0]"
1080 PRINT:PRINT:PRINT:PRINT
1090 GOSUB 1470
1100 GOTO 610
1110 REM ** MODIFY SELECTED QUESTION
1120 PRINT CHR$(147):PRINT TAB(240)
1130 M=1
1140 PRINT "ENTER QUESTION NUMBER TO BE MODIFIED"
1150 PRINT
1160 GOSUB 1520
1170 N=VAL(I$)
1180 PRINT CHR$(147)
1190 GOSUB 450
1200 GOTO 610
1210 REM ** DISPLAY QUESTIONS
1220 PRINT CHR$(147)
1230 FOR N=1 TO L
1240 PRINT CHR$(147)
1250 PRINT "[39↑0]"
1260 FOR Q=1 TO 7
1270 IF Q=1 THEN PRINT "[5 SPC]QUESTION NUMBER ";
1280 PRINT A$(N,Q)
1290 PRINT "[39↑F]"
1300 NEXT
1310 PRINT:PRINT:PRINT
1320 GOSUB 1470
1330 PRINT
1340 NEXT
1350 GOTO 610
1360 REM ** LOAD NEW TAPE
1370 PRINT CHR$(147):PRINT TAB(240)
1380 PRINT "  1. PLACE THE DATA TAPE IN CASSETTE"
1390 PRINT
1400 PRINT "  2. REWIND IT"
1410 PRINT
1420 PRINT "  3. SWITCH THE MOTOR OFF"
1430 PRINT:PRINT:PRINT
1440 GOSUB 1470
1450 PRINT CHR$(147)
1460 GOTO 1860
1470 REM ** PRESS-KEY
1480 PRINT "  PRESS ANY KEY WHEN READY TO PROCEED"
1490 GET W$
1500 IF W$="" THEN 1490
1510 RETURN
1520 REM ** CP INPUT
1530 OPEN 1,0
1540 INPUT#1,I$
1550 IF I$="" THEN 1540
1560 CLOSE 1
1570 RETURN
1580 REM ** SATISFIED?
1590 PRINT "YOU NOW HAVE ";FRE(0);" BYTES LEFT"
1600 PRINT
1610 PRINT "[5 SPC]DO YOU WANT TO MODIFY?"
1620 GOSUB 1520
1630 IF I$<>"YES" THEN F=1
1640 RETURN
1650 REM ** PREPARE TO OPEN 'WRITE' FILE
1660 POKE 243,122:POKE 244,2
1670 RETURN
1680 REM ** PREPARE TO CLOSE WRITE
1690 IF PEEK(625)>100 THEN 1710
1700 RETURN
1710 POKE 59411,53
1720 T8=TI
1730 IF TI-T8<6 THEN 1730
1740 POKE 59411,61
1750 RETURN
1760 DATA "   QUESTION NUMBER "
1770 DATA "ENTER THE QUESTION"
1780 DATA "ENTER THE FIRST ANSWER  (A)"
1790 DATA "ENTER THE SECOND ANSWER (B)"
1800 DATA "ENTER THE THIRD ANSWER  (C)"
1810 DATA "ENTER THE FOURTH ANSWER (D)"
1820 DATA "WHICH ANSWER A,B,C,D IS CORRECT ?"
1830 GOSUB 1520
1840 REM ** TEST TAPE RUNBACK
1850 PRINT CHR$(147)
1860 OPEN 1,1,0
1870 PRINT "WARNING !  TAPE DELIVERY IS IN BURSTS."
1880 PRINT:PRINT
1890 PRINT "JUST HAVE PATIENCE AND OBEY ORDERS"
1900 PRINT:PRINT:PRINT
1910 FOR Q=0 TO 4
```

modifications to the program itself may be necessary in some cases. Thus the number of questions are fixed at 25 but can be changed by altering the value of 'L' in line 100 of the PREPARATION program. Only those with 16K PETs however should increase L to say 50 or 100 because of the possibility of 'OUT OF MEMORY ERROR' More than one copy of the questions DATA tape can be made by simply using option '5' which is 'LOAD AN EXISTING TAPE' insert a blank tape and then use option '4' to 'SAVE QUESTIONS ON TAPE'.

```
1920 INPUT#1,A$(0,Q)
1930 PRINT A$(0,Q)
1940 NEXT
1950 GOSUB 1470
1960 FOR N=1 TO L
1970 FOR Q=1 TO 7
1980 INPUT#1,A$(N,Q)
1990 PRINT A$(N,Q)
2000 NEXT
2010 PRINT
2020 NEXT
2030 CLOSE 1
2040 GOSUB 1470
2050 PRINT CHR$(147):PRINT TAB(240)
2060 PRINT "TAPE DATA IS NOW LOADED"
2070 PRINT "[23↑0]"
2080 PRINT:PRINT:PRINT:PRINT
2090 GOSUB 1470
2100 GOTO 610
```

Listing 1. The data preparation program used by the teacher or lecturer to produce the series of questions and their possible answers.

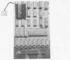
```
100 DIM A$(25,7)
110 L=25:G=1:H=3600
120 PRINT CHR$(147)
130 PRINT "[7 SPC]MULTIPLE CHOICE EXAM"
140 PRINT
150 PRINT " THIS PROGRAM REQUIRES A DATA TAPE WITH"
160 PRINT
170 PRINT "    THE QUESTIONS STORED ON IT !"
180 PRINT
190 PRINT "  1. PLACE THIS TAPE IN THE CASSETTE"
200 PRINT
210 PRINT "  2. REWIND IT"
220 PRINT
230 PRINT "  3. SWITCH THE MOTOR OFF"
240 PRINT:PRINT
250 GOSUB 1490
260 PRINT CHR$(147)
270 PRINT:PRINT
280 PRINT "[39↑0]"
290 PRINT "OBEY THE FOLLOWING AND RELAX FOR A BIT!"
300 PRINT "[39↑0]"
310 PRINT:PRINT:PRINT:PRINT
320 OPEN 1,1,0
330 FOR Q=0 TO 4
340 INPUT#1,A$(0,Q)
350 GOSUB 1530
360 NEXT
370 REM ** INPUT MAIN DATA
380 FOR N=1 TO L
390 FOR Q=1 TO 7
400 INPUT#1,A$(N,Q)
410 GOSUB 1530
420 NEXT:NEXT
430 CLOSE 1
440 PRINT CHR$(147):PRINT TAB(240)
450 PRINT "THE QUESTIONS ARE NOW LOADED"
460 GOSUB 1490
470 PRINT CHR$(147)
480 PRINT:PRINT
490 PRINT "ENTER YOUR NAME AND INITIALS"
500 GOSUB 1630
510 N$=I$:PRINT
520 PRINT "[39↑0]"
530 PRINT
540 PRINT "WHEN THE QUESTIONS APPEAR,KEY EITHER"
550 PRINT
560 PRINT "A,B,C OR D.  OTHER KEYS IMPLY [REV]PASS[OFF]"
570 PRINT:PRINT
580 GOSUB 1490
590 PRINT CHR$(147)
600 P=0:R=0:N1=0
610 PRINT "      25 MULTIPLE CHOICE QUESTIONS"
620 PRINT
630 PRINT
640 PRINT " LIBRARY TITLE ..";A$(0,0)
650 PRINT
660 PRINT " DATE OF COMPILATION ..";A$(0,2)
670 PRINT
680 PRINT " COMPILED BY ..";A$(0,1)
690 PRINT
700 PRINT " MINIMUM PASS MARK ..";A$(0,3)
710 PRINT
720 PRINT " TIME ALLOWED ..";A$(0,4);" MINUTES"
730 PRINT
740 PRINT
750 PRINT "[9 SPC][REV]TIME CLOCK STARTS[OFF]"
760 PRINT:PRINT:PRINT:PRINT
770 GOSUB 1490
780 TI$="000000"
790 FOR N=1 TO L
800 PRINT CHR$(147)
810 PRINT "[30↑0]"
820 PRINT "[10 SPC]QUESTION NUMBER ";N
830 PRINT "[30↑F]"
840 PRINT "TIME  ";
850 GOSUB 1570
860 PRINT "[30↑0]"
870 PRINT
880 FOR Q=2 TO 7
890 IF Q=7 THEN 950
900 K=62+Q:IF Q=2 THEN K=0
910 IF Q=2 THEN K=0
920 PRINT A$(N,Q)TAB(38)CHR$(K)
930 PRINT
940 GOTO 1060
950 PRINT "[39↑0]"
960 PRINT "ENTER A SINGLE LETTER,A OR B OR C OR D."
970 PRINT
980 PRINT "THINK CAREFULLY!   IF YOU CHANGE YOUR"
990 PRINT "MIND, [REV]DELETE[OFF]....AND RE-ENTER LETTER."
1000 PRINT
1010 PRINT "WHEN SATISFIED,PRESS [REV]RETURN[OFF] KEY"
1020 GOSUB 1630
1030 IF I$=A$(N,7) THEN R=R+1
1040 IF I$<>"A" AND I$<>"B" AND I$<>"C" AND I$<>"D" THEN N1=N1+1
1050 GOTO 1070
1060 IF Q=2 THEN PRINT "[39↑F]"
1070 NEXT
1080 REM ** TIME OUT
1090 IF TI> H*VAL(A$(0,4)) THEN 1120
1100 NEXT
1110 GOTO 1180
1120 PRINT CHR$(147)
1130 PRINT:PRINT:PRINT:PRINT
1140 PRINT "YOU HAVE EXCEEDED TIME ALLOWED"
1150 PRINT "[30↑0]"
1160 PRINT:PRINT:PRINT
1170 GOSUB 1490
1180 PRINT CHR$(147)
1190 PRINT "STUDENT'S NAME ..";N$
1200 PRINT "[14↑0]"
1210 P=R*100/L:P=INT(10*P+0.5)/10
1220 IF P<VAL(A$(0,3)) THEN G$="FAIL"
1230 IF P>=VAL(A$(0,3)) AND P<80 THEN G$="PASS"
1240 IF P>=80 AND P<90 THEN G$="CREDIT"
1250 IF P>=90 THEN G$="DISTINCTION"
1260 PRINT
1270 PRINT "[9 SPC]PERFORMANCE ANALYSIS"
1280 PRINT:PRINT
1290 PRINT "NUMBER OF RESPONSES ......"TAB(33)N-1
1300 PRINT
1310 PRINT "RIGHT ANSWERS ......"TAB(33)R
1320 PRINT
1330 PRINT "NUMBER OF PASSES........"TAB(33)N1
1340 PRINT
1350 PRINT "PERCENTAGE RECORDED ..........";P
1360 PRINT
1370 PRINT "GRADING ................";G$
1380 PRINT:PRINT
1390 PRINT "THIS IS YOUR ATTEMPT NUMBER ";G
1400 PRINT "[30↑0]"
1410 PRINT
1420 PRINT "HAVE YOU PERMISSION FOR ANOTHER ATTEMPT"
1430 PRINT
1440 PRINT "[5 SPC]ANSWER Y(YES) OR N(NO)"
1450 GET K$
1460 IF K$="" THEN 1450
1470 IF K$="Y" THEN G=G+1:GOTO 590
1480 GOTO 1180
1490 PRINT "  PRESS ANY KEY WHEN YOU ARE READY"
1500 GET K$
1510 IF K$="" THEN 1500
1520 RETURN
1530 IF(ST)=0 OR (ST)=64 OR (ST)=-128 THEN 1560
1540 PRINT "[8 SPC][REV]TAPE STATUS ERROR[OFF]"
1550 STOP
1560 RETURN
1570 H$=LEFT$(TI$,2):M$=MID$(TI$,3,2):S$=RIGHT$(TI$,2)
1580 IF H$="00" THEN 1600
1590 PRINT " ";H$;" HOURS ";
1600 PRINT TAB(18) M$;" MINUTES ";S$;" SECONDS
1610 RETURN
1620 REM ** CRASH-PROOF INPUT
1630 OPEN 1,0
1640 INPUT#1,I$
1650 IF I$="" THEN 1640
1660 CLOSE 1
1670 RETURN
```

Listing 2. The examination program uses the prepared data tape, regardless of the type of subject under examination.

Because of the limited characters per line and number of lines on the PET screen, the following rules apply to preparing questions:
The question must be limited to TWO lines. Remember to use the Space key to turn the corner to the second line . . .not the Return key. Each answer must be limited to one line.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

8032 UPGRADE



64K MEMORY EXPANSION

- 1 year unconditional warranty
- Compatible with all 8096 software
- Does not obstruct ROM sockets
- Fits SK computer
- No modification to 8032 required
- End User Price £ 400 + VAT
- Generous Dealer Discounts available

SOFTWARE (UK) Ltd.

RAGLAN HOUSE , 56 LONG ST , DURSLEY ,
Gloucestershire . Tel: 0453 46065

SM – TEXT

PROFESSIONAL TEXT PROCESSING FOR THE COMMODORE

MULTI – COLUMN – FORM MODE – STANDARD LETTERS
INVOICING – NOTED TEXT – BLOCK TRANSPORT

8032 or **8096** End User Price £ 400
Dealer Discounts for all products on request –
Sales Literature , Posters , etc . available f.o.c .

SM – CUDA Customer Database . A complete solution to Customer Administration problems
Versatile simplicity !!

LOS – 96 From the creators of the 8096 Operating System – Programmers' Aid versions for BASIC and 6502 Assembler .

sirius 1 *Software available soon !!!*

SOFTWARE (UK)

SUBSCRIPTION ORDER FORM

Cut out and SEND TO :
Electronics Today International,
513, LONDON ROAD,
THORNTON HEATH,
SURREY,
ENGLAND.

Please commence my personal subscription to Electronics Today International with the .... issue

SUBSCRIPTION RATES
(tick ☐ as appropriate)

£13.15 for 12 issues UK ☐
£16.95 for 12 issues overseas surface ☐
£36.95 for 12 issues Air Mail ☐

I am enclosing my (delete as necessary) Cheque/Postal Order/International Money Order for

£..........

(made payable to A.S.P. Ltd)
OR
Debit my Access/Barclaycard*
(*delete as necessary)

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss) ....
delete accordingly
Address ....
....
....
Signature ....
Date ....

electronics today
FREE INSIDE
Put safety first with our earth leakage circuit breaker
Spectra column – a new type of disco lighting
Avoid awkward silences with our signal line checker
-PLUS-
Regular Features on designing with transistors and micro systems
EXTRA...EXTRA...
BREADBOARD EXHIBITION GUIDE
electronics today
BUYER'S GUIDE TO CONQUERING THE UNIVERSE:
VIDEO GAMES FROM BOTH SIDES OF THE SCREEN
PLUS
AUDIO COMPUTING MUSIC RADIO ROBOTICS

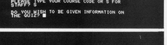
SUBSCRIPTIONS

The magazine you hold in your hand is the biggest seller in the UK electronics field. Why risk your newsagent running out? Take out a subscription using the form provided, and make sure of getting the next 12 issues. Don't you *deserve* not to miss out on the best?

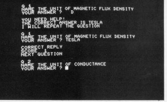
Len Topple

QUIZ TIME

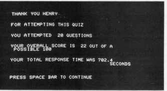
A multiple choice test program that's been proved in the classroom.



```
PLEASE TYPE YOUR NAME? HENRY

PLEASE TYPE YOUR COURSE CODE OR S FOR
STAFF? S

DO YOU WISH TO BE GIVEN INFORMATION ON
THE QUIZ? █
```



```
Q 1
NAME THE UNIT OF MAGNETIC FLUX DENSITY
YOUR ANSWER ?   D

YOU NEED HELP!
THE CORRECT ANSWER IS TESLA
I WILL REPEAT THE QUESTION

Q 1
NAME THE UNIT OF MAGNETIC FLUX DENSITY
YOUR ANSWER ? TESLA

CORRECT REPLY
SCORE  1
NEXT QUESTION

Q 2
NAME THE UNIT OF CONDUCTANCE
YOUR ANSWER ? █
```



```
THANK YOU HENRY

FOR ATTEMPTING THIS QUIZ

YOU ATTEMPTED  20 QUESTIONS

YOUR OVERALL SCORE IS  22 OUT OF A
 POSSIBLE 100

YOUR TOTAL RESPONSE TIME WAS 702.4
                                SECONDS

PRESS SPACE BAR TO CONTINUE
```

Some typical illustrations from the program. If you get a question wrong it is asked again to ensure that you understood the correct answer.

We all know that the SI unit of current is the Ampere but do you know the name of the unit of luminous intensity and are you sure you can spell the name of the unit of conductance correctly? If the answer is no, then your friendly computer will prompt you with the answer so that by the second or third re-run, your score should have increased significantly until eventually you achieve a perfect score. If you are a high scorer anyway you can still use the quiz to help sharpen your keyboard response in your attempt to obtain a fast time.

SCORING AND TIMING

Out of a possible 24 questions in this quiz, only 20 are asked at any one session and these are chosen and presented in a random order. This makes it impossible for you to obtain a high score and a fast time simply by remembering the order of the answers. A note is kept by the computer as each question is asked so that repetition is avoided; this is the function of the array A(). The method of scoring is as follows: you are allowed two attempts at each question and score five or three marks for a correct answer at your first or second attempt respectively. If you are wrong at your second go or if you reply with 'D' (for don't know) then you are given the correct answer, but are still asked the question again to help you to fix the answer in your mind. A correct reply at this stage is ▶

worth one mark but if you are careless and give the wrong answer you will incur a penalty of minus one!

While that explains the scoring of a normal run you may, if you wish, exit from the program by typing 'X' and for that question you will score minus three marks. After the normal 20 questions or after an 'X' reply you will be shown a list of the unasked questions so that you can test yourself on these as well but without the benefit of the model answers. There is one further penalty which will be incurred by anyone asleep at the keyboard. After a don't know reply to a question you are told the correct answer — so if you give a 'D' reply a second time to the same question, you will be given a penalty of minus five marks to help wake you up!

The normal total score will lie between 20 and 100 but you should get 90 or more after two runs. At this point your aim should be 100% in a fast time. The method of timing used in this quiz is to add up your individual response times to each question — the time taken by the computer to present each question is *not* included. The total time recorded is rounded to the nearest tenth of a second and you can use this, together with the score, to check your progress. If you have a printer then you can receive a certificate which includes a record of your score and speed.

The printer used to produce this certificate was the CBM 4022; you may have to modify subroutine SR5 if you use a different model. The printer is, of course, an optional extra in this program and may be omitted altogether by deleting lines 790 to 810 in the main program and deleting the whole of SR5.

A problem common to many interactive programs is that of a program crash which occurs if, in response to an INPUT statement, the Return key is hit before any other key. In a quiz this mistake would be very tiresome, particularly if the participant was half way through the questions. To prevent this crash, line 3060 has been modified and line 3070 added in SR3. If now, in response to 'YOUR ANSWER?', the Return key is hit then the underline will prevent a program crash. If, however, a normal answer is typed then the first character will replace the underline and a simple comparison with the model answer can take place.

CHANGING THE QUESTIONS

The questions in this SI quiz all require single word answers which must be spelt correctly. It is a simple task to modify the program so that questions are asked on an entirely different topic provided that you can keep to the same format of single word answers. An obvious alternative quiz of this type would include questions and answers such as:

WHAT IS THE CAPITAL OF FRANCE?

If you look at line 3040 you will see that each question is a part concatenation of a fixed part (Q1$) and a variable part (Q$). The variable part of the questions and the model answers are stored in the DATA statements which are very easy to change. If, for some reason, it is necessary to present the questions in a predetermined order then line 3030 together with SR6 should be deleted. It would, however, be a relatively simple matter to change the program so that one of several predetermined selections was chosen at random on each run.

Suppose, however, you wish to accept two or more correct answers. Clearly all the acceptable answers must be included in your DATA statements and the READ statement in line 6030 of SR6 must be changed as well as the test for the correct response in line 3100. If some of your questions have one correct answer while others have two or more then a further modification is necessary to allow for these variations. Including null strings in the data would be one of several ways of doing this.

A word for warning should be heeded by those of you who wish to create a new set of questions. Make sure that your questions are not ambiguous and also double check on the correctness of your model answers. It is a good idea to shelter behind the protection of an established authority on the

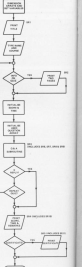


Fig. 1. The main flowchart.

subject. In the SI quiz, my reference is the well known book **Physical Constants** by Kaye and Laby (14th Edition).

THE PROGRAM

Even if you are not interested in this kind of quiz I hope that you may find some useful programming ideas. First, you must decide precisely what it is you wish the program to achieve. The outline scheme for this quiz, but not all of the detail, has already been described. Now examine the main program flowchart (Fig. 1) and you will see that most of the detailed programming is put into the subroutines. A subroutine is a module which can be designed and tested as an entity and used in other programs if required. The subroutine will operate in any program subject only to the correct transfer of data to and from the subroutine. This method of design is to be recommended because it is easier to achieve high reliability and good maintenance, that is, the ability to modify the program to meet new conditions with no or few faults. Debugging a new program is easier too!

It is highly desirable that the user should be given sufficient information on the screen to enable him to use the program correctly. On the other hand, not all of the information need be repeated on each re-run and a choice is given to the user, such as in line 700, which may save much frustration. The time taken to read a screen page of information varies considerably from user to user and depends, in part, on previous usage. For this reason, information should not be presented for a fixed time; instead, the user should be given a simple instruction which, when carried out, 'turns' the page. In this quiz the instruction used is 'PRESS SPACE BAR TO CONTINUE' but, as you will see from lines 1080 and 1090 in SR1, the program would continue if *any* key is pressed. The reason for giving the specific instruction rather than the more general 'PRESS ANY KEY' is that the latter may raise a doubt in the user's mind about the possibility of different effects resulting from pressing different keys.

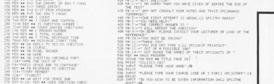
The three line press space bar routine occurs a number of times throughout the program and could have been turned into a four line subroutine which was called by a GOSUB on each occasion as required. The message structure and the control structure used in this program could also be simplified if only a single reply to each question is permitted. This, however, is more appropriate to a competitive type of quiz than to a self-learning scheme with rewards and penalties. The variable F is used both for control and for scoring (see SR3).

The variable T is used to sum the total time to reply to each question. This is set to zero at the beginning of a run (line 320) and is measured in 'jiffies' **('jiffies' are the internal divisions of time on the PET and represent 50ths of a second. Ed.)** while the increments of time are being added together. T is finally converted to seconds (rounded to the nearest tenth of a second) in line 3230.

The 4022 printer used in this quiz to produce a certificate has a wide variety of format controls. Use has been made of the control CHR$(1) in lines 5120, 5130, 5240 and 5370, to change the print size. Clearly these lines may have to be modified to suit the printer available to you.

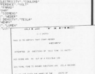
RUNNING THE PROGRAM

This is perfectly straightforward, provided that the user follows the screen instructions. Yes/No answers to questions such as do you wish to be given information on the quiz? may be given in full or abbreviated to Y or N. The program has been tested on many first year students who have used it without supervision. Reactions to the quiz have shown me that many students like this kind of learning process. It is, however wise to take note of the final message which is to remember that knowing the names of the units is *not* the same as understanding their meaning.



```
100 REM ** VARIABLES AND DIMENSIONS
110 D=0 REM ** NO. OF  DON'T KNOWS PER QUESTION
120 F=0 REM ** NO OF ERRORS
130 REM ** F=1 ONE ERROR
140 REM ** F=2 TWO ERRORS OR DON'T KNOW
150 REM ** F=3 THREE ERRORS
160 REM ** F=4 EXIT FROM PROGRAM
170 REM ** F=5 TWO DON'T KNOWS
180 I=0 REM ** I COUNT
190 J=0 REM ** J COUNT AND CONTROL
200 J1=0 REM ** J1 QUESTION NUMBER
210 J2=0 REM ** COUNT UNUSED QUESTIONS
220 K=0 REM ** K RANDOM NUMBER
230 S=0 REM ** S TOTAL SCORE
240 T=0 REM ** T TOTAL RESPONSE TIME
250 T0=0 REM ** RESPONSE START TIME
260 REM ** A$ REPLY TO YES/NO QUESTION
270 REM ** C$ COURSE.
280 REM ** M$ MODEL ANSWER
290 REM ** N$ NAME
300 REM ** O$ QUESTION VARIABLE PART
310 O1$="NAME THE UNIT OF "
320 P$="PRESS SPACE BAR TO CONTINUE"
330 REM ** R$ RESPONSE TO QUESTION
340 S1$="STAFF MEMBER"
350 S2$=" STUDENT"
360 REM ** W$ WAIT FOR SPACE BAR
370 DIM A(24) REM ** USED QUESTION CHECK
380 DIM M$(14) REM ** MESSAGES
390 DIM U$(24) REM ** UNUSED QUESTIONS
400 M$(1)="CORRECT REPLY"
410 M$(2)="NEXT QUESTION"
420 M$(3)="I AM SORRY THAT YOU HAVE GIVEN UP BEFORE THE END OF
    THE TEST"
430 M$(4)=" WHY NOT CONSULT YOUR NOTES AND TRY[5 SPC]AGAIN
    TOMORROW?"
440 M$(5)="YOUR FIRST ATTEMPT IS WRONG.[2 SPC]TRY AGAIN"
450 M$(6)="YOU NEED HELP!"
460 M$(7)="THE CORRECT ANSWER IS "
470 M$(8)="I WILL REPEAT THE QUESTION"
480 M$(9)="OH DEAR! PLEASE CONSULT YOUR LECTURER OR LOOK UP THE
    REFERENCE"
490 M$(10)="YOU MUST BE JOKING"
500 M$(11)="THANK YOU"
510 M$(12)="YOU DID NOT TYPE Y.[2 SPC]SKIP PRINTOUT"
520 M$(13)=" OUT OF A POSSIBLE 100"
530 M$(14)=" NOT ASKED THE NAMES OF THE[5 SPC]UNITS OF "
540 REM ** MAIN PROGRAM
550 GOSUB 780 REM ** TITLE PAGE SR1
560 PRINT "[CLS][3 CD]"
570 INPUT "PLEASE TYPE YOUR NAME";N$
580 PRINT
590 INPUT "PLEASE TYPE YOUR COURSE CODE OR S FOR[3 SPC]STAFF";C$
600 PRINT
610 INPUT "DO YOU WISH TO BE GIVEN INFORMATION ON[2 SPC]THE
    QUIZ";A$
```

```
620 IF LEFT$(A$,1)="Y" THEN GOSUB 900:REM ** SR2
630 S=0:T=0
640 FOR I=1 TO 24:REM ** INITIALISE
650 A(I)=0
660 NEXT I
670 GOSUB 1290:REM ** Q & A SR3
680 IF R$="X" THEN 710
690 INPUT "DO YOU WISH TO REPEAT THE QUIZ";A$
700 IF LEFT$(A$,1)="Y" THEN 610
710 GOSUB 1580:REM ** END SUBROUTINE SR4
720 PRINT
730 INPUT "DO YOU REQUIRE A CERTIFICATE OF YOUR[5 SPC]SCORE
    AND TIME";A$
740 IF LEFT$(A$,1)="Y" THEN GOSUB 1790:REM ** PRINTOUT SR5
750 PRINT:PRINT "END OF PROGRAM, GOODBYE"
760 END
770 REM ** TITLE SUBROUTINE SR1
780 PRINT "[CLS]"
790 PRINT "[5 CD][10 CR]";
800 PRINT "QUIZ TIME"
810 PRINT "[3 CD]"
820 PRINT "THIS IS A QUIZ ON THE NAMES OF SI UNITS"
830 PRINT
840 PRINT "USE THE QUIZ AS A TEST OF YOUR ACCURACY AND SPEED"
850 PRINT:PRINT:PRINT:PRINT P$
860 GET W$
870 IF W$="" THEN 860
880 RETURN
890 REM ** INFORMATION SUBROUTINE SR2
900 PRINT "[CLS]"
910 PRINT "IN THIS QUIZ YOU WILL BE ASKED 20"
920 PRINT "QUESTIONS OUT OF THE POSSIBLE 24"
930 PRINT:PRINT "THE QUESTIONS WILL BE PRESENTED TO YOU"
940 PRINT:PRINT "IN A RANDOM ORDER. YOU MUST ANSWER EACH"
950 PRINT:PRINT "QUESTION WITH A SINGLE WORD NAME"
960 PRINT:PRINT "YOUR SPELLING MUST BE CORRECT"
970 PRINT:PRINT:PRINT
980 PRINT "IF YOU DO NOT KNOW THE ANSWER TO A"
990 PRINT "QUESTION TYPE 'D'"
1000 PRINT:PRINT
1010 PRINT "IF YOU WISH TO END THE QUIZ TYPE X"
1020 PRINT:PRINT:PRINT
1030 PRINT P$
1040 GET W$
1050 IF W$="" THEN 1040
1060 PRINT "[CLS]"
1070 PRINT:PRINT:PRINT:PRINT
1080 PRINT "YOU WILL SCORE POINTS ON EACH QUESTION"
1090 PRINT "AS FOLLOWS"
1100 PRINT
1110 PRINT "CORRECT AT FIRST ATTEMPT ..............5"
1120 PRINT "CORRECT AT SECOND ATTEMPT .............3"
1130 PRINT "CORRECT AT THIRD ATTEMPT...............1"
1140 PRINT "WRONG AT THIRD AND FINAL ATTEMPT......-1"
1150 PRINT "PENALTY FOR PREMATURE ENDING OF TEST.-3"
1160 PRINT "PENALTY FOR TWO 'DONT KNOWS' IN"
1170 PRINT "[10 SPC]ONE QUESTION ......-5"
1180 PRINT:PRINT
1190 PRINT "MAXIMUM MARKS FOR 20 QUESTIONS  ....100
1200 PRINT
1210 PRINT "EACH QUESTION IS TIMED FROM THE END OF"
1220 PRINT "THE QUESTION PRINTOUT TO WHEN YOU HIT"
1230 PRINT "THE RETURN KEY"
1240 PRINT:PRINT:PRINT:PRINT P$
1250 GET W$
1260 IF W$="" THEN 1250
1270 RETURN
1280 REM ** Q & A SUBROUTINE SR3
1290 PRINT "[CLS]"
1300 FOR J=1 TO 20
1310 D=0:F=0:J1=J:REM ** INITIALISE QUESTION
1320 GOSUB 2210:REM ** READ DATA IN RANDOM ORDER SR6
1330 PRINT:PRINT "Q";J1
1340 PRINT Q1$+Q$
1350 T0=TI
1360 INPUT "YOUR ANSWER [2 CR]_[3 CL]";R$
1370 IF R$="_" THEN PRINT "[2 CU]":GOTO 1360
1380 T=T+TI-T0
1390 PRINT
1400 IF R$=M$ THEN PRINT M$(1):PRINT "SCORE ";5-2*F:PRINT
     M$(2):GOTO 1480
1410 F=F+1
1420 IF R$="X" THEN GOSUB 2300:REM ** SR7
1430 IF R$="D" THEN GOSUB 2350:REM ** SR8
1440 IF F=1 THEN PRINT M$(5):GOTO 1330
1450 IF F=2 THEN PRINT M$(6):PRINT M$(7)+M$:PRINT M$(8):GOTO 1330
1460 IF F=3 THEN PRINT M$(9):PRINT "SCORE -1"
1470 IF F=5 THEN PRINT M$(10):PRINT "SCORE -5"
1480 S=S+5-2*F
1490 NEXT J
1500 PRINT:PRINT P$
1510 GET W$
1520 IF W$="" THEN 1510
1530 T=INT(T/6+.5)/10
1540 GOSUB 2400:REM ** STORE UNUSED QUESTIONS SR9
1550 IF F=4 THEN J1=J1-1
1560 RETURN
1570 REM ** END SUBROUTINE SR4
1580 PRINT "[CLS]"
1590 PRINT:PRINT:PRINT
1600 PRINT "THANK YOU ";N$
1610 PRINT:PRINT
1620 PRINT "FOR ATTEMPTING THIS QUIZ"
1630 PRINT:PRINT
1640 PRINT "YOU ATTEMPTED ";J1;"QUESTIONS"
1650 PRINT:PRINT
1660 PRINT "YOUR OVERALL SCORE IS ";S;"OUT OF A"
1670 PRINT "POSSIBLE 100"
1680 PRINT:PRINT
1690 PRINT "YOUR TOTAL RESPONSE TIME WAS";T;"SECONDS"
1700 GOSUB 2490:REM ** PRINT LIST UNUSED QUESTIONS SR10
1710 PRINT:PRINT
1720 PRINT "REMEMBER THAT KNOWING THE NAMES OF THE"
1730 PRINT "UNITS IS NOT THE SAME AS UNDERSTANDING"
1740 PRINT "THEIR MEANING"
1750 PRINT:PRINT
1760 PRINT "I HOPE THAT YOU ARE HAPPY, WISER OR BOTH"
1770 RETURN
1780 REM ** PRINT CERTIFICATE SR5
1790 PRINT "[CLS][3 CD]"
1800 PRINT "IS YOUR PRINTER CONNECTED,LOADED WITH"
1810 PRINT "PAPER AND SWITCHED ON?"
1820 PRINT
1830 PRINT "TYPE Y WHEN READY"
1840 GET W$
1850 IF W$="" THEN 1840
1860 IF W$<>"Y" THEN PRINT M$(12):GOTO 2190
1890 FOR I=1 TO 10
1900 PRINT#1,CHR$(29);
1910 NEXT I
1920 PRINT#1,CHR$(1);"Q";CHR$(1);"U";CHR$(1);"I";CHR$(1);"Z";CHR$(29);
1930 PRINT#1,"T";CHR$(1);"I";CHR$(1);"M";CHR$(1);"E"
1940 PRINT#1:PRINT#1
1950 FOR I=1 TO 20
1960 PRINT#1,CHR$(29);
1970 NEXT I
1980 PRINT#1,"(S I UNITS)"
1990 PRINT#1:PRINT#1
2000 PRINT#1,"THIS IS TO CERTIFY THAT ";
2010 IF C$="S" THEN PRINT#1,S1$:GOTO 2030
2020 PRINT#1,C$;S2$;" "
2030 PRINT#1
2040 PRINT#1,CHR$(29);CHR$(1);N$
2050 PRINT#1:PRINT#1:PRINT#1
2060 PRINT#1," ATTEMPTED ";J1;" QUESTIONS OF 'QUIZ TIME (SI UNITS)'"
2070 PRINT#1:PRINT#1:PRINT#1
2080 PRINT#1,"HIS SCORE WAS ";S;M$(13)
2090 PRINT#1:PRINT#1:PRINT#1
2100 PRINT#1,"HIS TOTAL TIME TO ANSWER QUESTIONS WAS ";T;"SECONDS"
2110 PRINT#1:PRINT#1:PRINT#1
2120 GOSUB 2590:REM ** UNUSED QUEST.(SR11)
2130 PRINT#1:PRINT#1:PRINT#1
2140 FOR I=1 TO 35
2150 PRINT#1,CHR$(29);
2160 NEXT I
2170 PRINT#1,"(SIGNED)";CHR$(29);CHR$(1);"O T";CHR$(29);CHR$(1);"PET"
2180 CLOSE1
2190 RETURN
2200 REM ** READ DATA IN RANDOM ORDER (SEE SR3)  SR6
2210 K=INT(24*RND(0))+1
2220 IF A(K)=1 THEN 2210
2230 FOR I=1 TO K
2240 READ Q$,M$
2250 NEXT I
2260 A(K)=1
2270 RESTORE
2280 RETURN
2290 REM ** EXIT (SEE SR3)  SR7
2300 PRINT "SCORE  -3"
2310 J=20:F=4
2320 PRINT M$(3):PRINT:PRINT M$(4)
2330 RETURN
2340 REM ** DON'T KNOW (SEE SR3)  SR8
2350 F=2
2360 IF D=1 THEN F=5
2370 D=D+1
2380 RETURN
2390 REM ** UNUSED QUESTIONS (SEE SR3) SR9
2400 J=1
2410 FOR K=1 TO 24
2420 READ Q$,M$
2430 IF A(K)=0 THEN U$(J)=Q$:J=J+1
2440 NEXT K
2450 J2=J
2460 RESTORE
2470 RETURN
2480 REM ** PRINT UNUSED QUESTIONS (SEE SR4) SR10
2490 PRINT:PRINT
2500 PRINT P$
2510 GET W$
2520 IF W$="" THEN 2510
2530 PRINT:PRINT "YOU WERE";M$(14)
2540 FOR J=1 TO J2
2550 PRINT U$(J)
2560 NEXT J
2570 RETURN
2580 REM ** PRINT# UNUSED QUESTIONS SR11 (SEE SR5)
2590 PRINT#1,"HE WAS";M$(14)
2600 FOR J=1 TO J2
2610 PRINT#1,U$(J)
2620 NEXT J
2630 RETURN
2640 REM ** DATA
2650 DATA "LENGTH","METRE"
2660 DATA "MASS","KILOGRAM"
2670 DATA "TIME","SECOND"
2680 DATA "ELECTRIC CURRENT","AMPERE"
2690 DATA "THERMODYNAMIC TEMPERATURE","KELVIN"
2700 DATA "LUMINOUS INTENSITY","CANDELA"
2710 DATA "AMOUNT OF SUBSTANCE","MOLE"
2720 DATA "PLANE ANGLE","RADIAN"
2730 DATA "SOLID ANGLE","STERADIAN"
2740 DATA "FREQUENCY","HERTZ"
2750 DATA "FORCE","NEWTON"
2760 DATA "PRESSURE","PASCAL"
2770 DATA "ENERGY","JOULE"
2780 DATA "POWER","WATT"
2790 DATA "QUANTITY OF ELECTRICITY","COULOMB"
2800 DATA "POTENTIAL DIFFERENCE","VOLT"
2810 DATA "CAPACITANCE","FARAD"
2820 DATA "RESISTANCE","OHM"
2830 DATA "CONDUCTANCE","SIEMENS"
2840 DATA "MAGNETIC FLUX","WEBER"
2850 DATA "MAGNETIC FLUX DENSITY","TESLA"
2860 DATA "INDUCTANCE","HENRY"
2870 DATA "LUMINOUS FLUX","LUMEN"
2880 DATA "ILLUMINANCE","LUX"
```



```
QUIZ  TIME

(S I UNITS)

THIS IS TO CERTIFY THAT STAFF MEMBER
HENRY

ATTEMPTED  20  QUESTIONS OF 'QUIZ TIME (SI UNITS)'

HIS SCORE WAS  90  OUT OF A POSSIBLE 100

HIS TOTAL TIME TO ANSWER QUESTIONS WAS  152.4 SECONDS

HE WAS NOT ASKED THE NAMES OF THE      UNITS OF
LUMINOUS INTENSITY
AMOUNT OF SUBSTANCE
FREQUENCY
RESISTANCE

GNED)                    O T PET
```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

APPLICATIONS

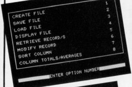
Mutlipurpose Records — Set up your own filing system which enables you to store, search, edit and retrieve data.
Originally published in **Computing Today**, August 1980.

VIC Editor — Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.
Originally published in **Computing Today**, November 1982.

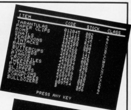
Commodore Communications — Get Commodores talking to each other using this application.
Originally published in **Computing Today**, November 1982.

Address Book — Compile an address book, or any other similar list, and throw away those bits of paper.

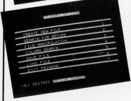
Multicolumn Records — A multipurpose data base program for use at home or in the office.
Originally published in **Computing Today**, July 1981.



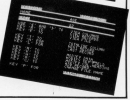
```
CREATE FILE                1
SAVE FILE                  2
LOAD FILE                  3
DISPLAY FILE               4
RETRIEVE RECORD/S          5
MODIFY RECORD              6
SORT COLUMN                7
COLUMN TOTALS/AVERAGES     8

       ENTER OPTION NUMBER
```



```
ITEM
                 CODE     STOCK   CLASS
TARANTULAS       012045   002     Z
PAPER CLIPS
ROMS
SNAILS
TEASPOONS
TINTACKS
RAMS
BUTTERCUPS
CONCORDS
PIANOS
CROCODILES
TWEEZERS
PYTHONS
SLIPPERS
BUNGALOWS
CORNFLAKES
BULLDOZERS

        PRESS ANY KEY
```



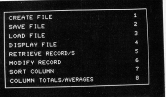
```
         PRIMARY OPTIONS

CREATE NEW FILE            1
SEARCH FOR RECORD          2
FILE MANIPULATION          3
COLUMN SEARCH              4
SAVE FILE                  5
LOAD FILE                  6
EXIT PROGRAM               7

KEY DESIRED OPTION NUMBER
```



```
NAME
                    AGE
                    22
USE '<' AND '>' TO
KEY 'R' TO          VIEW COLUMNS
KEY 'F' TO          VIEW RECORDS
                    VIEW FILE
KEY 'T' TO
KEY 'N' FOR         TOTALISE COLUMN
KEY 'L' FOR         NEXT RECORD
                    LAST RECORD
KEY 'M' TO
KEY 'K' TO          MODIFY DATA
KEY 'H' TO          MODIFY KEY FIELD
KEY 'A' TO          MODIFY COL.HEADING
KEY 'C' TO          ADD RECORD
                    CHANGE FILE NAME
KEY 'P' FOR         PRIMARY OPTIONS
```

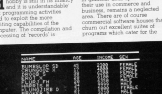
MULTI-PURPOSE RECORDS

Computers were meant to handle information and this program lets them do it in any environment, from the business to the home.



```
CREATE FILE                      1
SAVE FILE                        2
LOAD FILE                        3
DISPLAY FILE                     4
RETRIEVE RECORD/S                5
MODIFY RECORD                    6
SORT COLUMN                      7
COLUMN TOTALS/AVERAGES           8

          ENTER OPTION NUMBER
```

Home computing as a hobby is still in its infancy and it is understandable that programming activities tend to exploit the more exciting capabilities of the computer. The compilation and processing of 'records' is unfortunately a subject lacking in glamour and, apart from their use in commerce and business, remains a neglected area. There are of course commercial software houses that churn out excellent suites of programs which cater for the businessman but, from the small system users' viewpoint, they tend to suffer from high cost and inevitably require some form of disc backing store to cater for the vociferous appetites of the modern Accountants. At a guess, it is doubtful if more than one in twenty owners of a microcomputer have sufficiently recovered from the trauma of its initial purchase to even consider further expenditure on the floppy! Commercial software also assumes that facilities for hard copy output are available and refuses to cater for the underprivileged classes who don't have a printer. Thus the combination of cost, and lack of glamour has caused most enthusiasts to neglect the records area altogether. Now this is a pity. The ability of a computer to store, retrieve, select, modify and rearrange data is its most endearing quality.



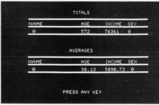
```
NAME                 AGE     INCOME  SEX
BUMBSLOP SD          45      2300    FEMALE
CRUNGOOZ X           17      4000    FEMALE
DROOB S              38      6200    FEMALE
EROT X               28      6100    FEMALE
FLURJ FH             20      6350    MALE
HARRIS R             34      9999    MALE
KLUNGJIRB D          31      5000    FEMALE
PROKER D             51      3210    MALE
ROBBO R              52      9000    MALE
ROTBLUG R            30      6000    MALE
SMITH A              23      5000    MALE
SMODDLE F            56      9800    MALE
STEPHENSON AP        99      1002    MALE
THATCHER M           23      COR!    FEMALE
ZLOBBLE SF           25      2400    MALE

             PRESS ANY KEY
```

A typical business use for the program would be to create an employee record.

TERMINOLOGY DEFINED

A few terms need to be defined before programs can be understood. A FILE is a collection of information on one subject; for example, a file name could be 'Clients Credit-worthiness' or 'Lepidoptera' or 'Power Transistors', 'World Religions' etc, etc. Files are normally displayed as a number of rows and columns. A RECORD is any individual ROW and a DATA-ITEM is any individual column in that row.

The minimum requirements of any program which assumes the title of General Purpose Records must allow the



```
             TOTALS

NAME              AGE    INCOME  SEX
 0                572    76361   0

            AVERAGES

NAME              AGE    INCOME  SEX
 0                38.13  5090.73  0



          PRESS ANY KEY
```

The inbuilt arithmetic functions allow us to perform calculations on records.

following options:

a) ***Create a new file*** by naming the column headings and entering the records from the keyboard.

b) ***Save the file*** on backing store under an appropriate file name.

c) ***Load a file*** from backing store.

d) ***Display the contents*** on the screen, page by page.

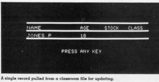
e) ***Retrieve a selected record or records*** on inputting a 'search key'.

f) ***Modify*** a selected record.

g) ***Sort*** the order in which the records are arranged under a selected column heading.

h) ***Calculate*** the sum, averages etc of the columns (where applicable).

This program attemps to satisfy the minimum requirements and subject to the limitations imposed by small memory, will just squeeze itself into an 8K PET providing the number of rows (size of the file) is kept to a modest level. With a 16K PET there is ample room for the display and processing of several hundred records and apart from the slowness of cassette tape access, will satisfy a practical need.



```
NAME              AGE    STOCK   CLASS
JONES P           18


          PRESS ANY KEY
```

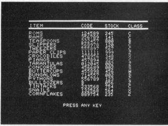
A single record pulled from a classroom file for updating.

PROGRAMMING NOTATION

To get over the problems of not being able to reproduce the graphics characters produced by PET **Computing Today** standards are followed.

VISUAL RESTRICTION

Unfortunately, the PET screen is only 40 characters wide which restricts the number of columns in a record to about four. Even then, the column headings must be entered with restraints on the number of characters. A relatively wide field has been allowed for column one because this is normally the key to the record, such as the name of a person, animal, insect or object. If a data item is considered long the program warns of this but accepts it. Data is held in the string array A (R,C) and is dimensioned in line 10 as (100,5) which allows for a nominal 100 rows and 5 columns. For 8K models, the dimensioning is too ambitious and will need reducing to say 25,5 in order to avoid that awful 'OUT OF MEMORY ERROR'. For 16K PETs on the other hand, the dimensions are too modest and can be increased to over 200 rows if desired.



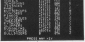
```
ITEM              CODE    STOCK   CLASS
ROMS              124589  245     C
RAMS              356789  141     C
TEASPOONS         222457  800     B
SLIPPERS          568978  120     C
TWEEZERS          555897  567     C
PAPER CLIPS       111111  999     Z
CROCODILES        478956  348     X
PIANOS            457894  325     C
TARANTULAS        012045  002     Z
CONCORDS          456689  002     S
BUTTERCUPS        455898  456     W
BUNGALOWS         714557  002     D
PYTHONS           556789  123     C
BULLDOZERS        B       056     Z
TINTACKS          233568  754     C
SNAILS            132568  564     X
CORNFLAKES        889754  235     Z

           PRESS ANY KEY
```

Quite what shop or store would stock such a diverse range is a mystery but this is another typical area of application for the program.

```
100 DIM A$(25,5)
110 PRINT CHR$(147)
120 PRINT "[↑O][37↑*][↑P]"
130 PRINT "[↑%]CREATE FILE[23 SPC]1[2 SPC][↑/]"
140 PRINT "[↑%][37 SPC][↑/]"
150 PRINT "[↑%]SAVE FILE[25 SPC]2[2 SPC][↑/]"
160 PRINT "[↑%][37 SPC][↑/]"
170 PRINT "[↑%]LOAD FILE[25 SPC]3[2 SPC][↑/]"
180 PRINT "[↑%][37 SPC][↑/]"
190 PRINT "[↑%]DISPLAY FILE[22 SPC]4[2 SPC][↑/]"
200 PRINT "[↑%][37 SPC][↑/]"
210 PRINT "[↑%]RETRIEVE RECORD/S[17 SPC]5[2 SPC][↑/]"
220 PRINT "[↑%][37 SPC][↑/]"
230 PRINT "[↑%]MODIFY RECORD[21 SPC]6[2 SPC][↑/]"
240 PRINT "[↑%][37 SPC][↑/]"
250 PRINT "[↑%]SORT COLUMN[23 SPC]7[2 SPC][↑/]"
260 PRINT "[↑%][37 SPC][↑/]"
270 PRINT "[↑%]COLUMN TOTALS/AVERAGES[12 SPC]8[2 SPC][↑/]"
280 PRINT "[↑L][37↑*][↑/]"
290 PRINT:PRINT:PRINT
300 PRINT "[REV][10 SPC][OFF]ENTER OPTION NUMBER[REV][10 SPC][OFF]"
310 GET K$
320 IF K$="" THEN 310
330 E$="INPUT UNACCEPTABLE"
340 IF VAL(K$)<1 OR VAL(K$)>8 THEN PRINT E$:PRINT:PRINT:PRINT:GOTO 290
350 ON VAL(K$)GOTO 360,860,1250,2180,2230,2610,1590,1840
360 REM ** INPUT LIMITS
370 R$="LIMIT IS 100!":C$="LIMIT IS 4!"
380 PRINT CHR$(147):PRINT TAB(240)
390 PRINT "HOW MANY RECORDS?";TAB(25);
400 GOSUB 2840
410 PRINT
420 Y=VAL(A$(R,C))
430 PRINT
440 IF Y<1 OR Y>100 THEN PRINT R$:PRINT:PRINT:GOTO 390
450 PRINT "HOW MANY COLUMNS?";TAB(25);
460 GOSUB 2840
470 PRINT
480 X=VAL(A$(R,C))
490 PRINT
500 IF X<1 OR X>4 THEN PRINT C$:PRINT:PRINT:GOTO 450
510 PRINT CHR$(147):M=1
520 PRINT "GIVE TITLE OF COLUMN";M;TAB(25);
530 GOSUB 2840
540 PRINT:PRINT
550 C1$(M)=A$(R,C)
560 T1$=C1$(1)
570 IF M=X THEN 600
580 M=M+1
590 GOTO 520
600 REM ** INPUTTING ENTIRE RECORD LIST
610 PRINT CHR$(147)
620 PRINT "[39↑8]"
630 PRINT "       COMMENCE ENTERING RECORDS"
635 PRINT
640 PRINT "      PRESS [REV]RETURN[OFF] AFTER EACH ENTRY"
650 PRINT
660 PRINT "[39↑8]"
670 W$="[REV]WARNING[OFF]. DATA ACCEPTED.INCONVENIENT."
680 IR$="ENTER "
690 FOR R=1 TO Y
700 FOR C=1 TO X
710 IF C<>1 THEN 760
720 PRINT IR$;T1$;TAB(25);
730 GOSUB 2840
740 PRINT
750 GOTO 810
760 PRINT:PRINT IR$;C1$(C);TAB(25);
770 GOSUB 2840
780 PRINT
790 IF LEN(A$(R,C))<7 THEN 810
800 PRINT:PRINT W$
810 NEXT
820 PRINT
830 PRINT "[39↑8]"
840 NEXT
850 GOTO 110
860 REM ** SAVE FILE
870 PRINT CHR$(147):PRINT TAB(240):T=2
880 PRINT:PRINT"ENTER FILE NAME";TAB(18);
890 GOSUB 2840
900 PRINT CHR$(147):PRINT TAB(240)
910 PRINT"WILL THIS BE LAST FILE ON THIS TAPE?
920 PRINT"[7 SPC]ANSWER Y(YES) OR N(NO)"
930 PRINT
940 GET K$
950 IF K$="" THEN 940
960 IF K$="N" THEN T=1
970 PRINT TAB(120)
980 PRINT "HAVE YOU INSERTED A CASSETTE?"
990 PRINT
1000 PRINT "HAVE YOU REWOUND IT TO DESIRED POSITION"
1010 PRINT
1020 PRINT "HAVE YOU SWITCHED MOTOR OFF?"
1030 PRINT:PRINT
1040 PRINT"[REV]YOU MUST NOW ANSWER Y(YES)[OFF]"
1050 GET K$
1060 IF K$<>"Y" THEN 1050
1070 F$=A$(R,C)
1080 PRINT CHR$(147):PRINT TAB(240)
1090 GOSUB 2030
1100 OPEN 1,1,T,F$
1110 PRINT#1,Y:PRINT#1,X
1120 FOR R=1 TO Y
1130 FOR C=1 TO X
1140 PRINT#1,A$(R,C)
1150 NEXT C
1160 NEXT R
1170 FOR H=1 TO X
1180 PRINT#1,C1$(H)
1190 NEXT
1200 GOSUB 2060
1210 CLOSE 1
1220 PRINT:PRINT:PRINT "DATA NOW ON TAPE
1230 GOSUB 3080
1240 GOTO 110
1250 PRINT CHR$(147):PRINT:REM ** LOAD TAPE
1260 PRINT "ENTER FILE NAME";TAB(19);
1270 GOSUB 2840
1280 PRINT
1290 PRINT TAB(240)
1300 PRINT "IS THIS FILE IN CASSETTE?"
1310 PRINT
1320 PRINT "IS IT REWOUND?"
1330 PRINT
1340 PRINT "IS THE MOTOR SWITCHED OFF?"
1350 PRINT:PRINT:PRINT
1360 PRINT "IF SO,ANSWER Y(YES)"
1370 GET H$
1380 IF H$="" THEN 1370
1390 IF H$<>"Y" THEN 1370
1400 PRINT CHR$(147):PRINT TAB(240)
1410 F$=A$(R,C)
1420 OPEN 1,1,0,F$
1430 INPUT#1,Y$:INPUT#1,X$
1440 GOSUB 2130
1450 Y=VAL(Y$):X=VAL(X$)
1460 FOR R=1 TO Y
1470 FOR C=1 TO X
1480 INPUT#1,A$(R,C)
1490 GOSUB 2130
1500 NEXT C
1510 NEXT R
1520 FOR H=1 TO X
1530 INPUT#1,C1$(H)
1540 GOSUB 2130
1550 NEXT
1560 CLOSE 1
1570 GOSUB 3080
1580 GOTO 110
1590 REM ** SORT
1600 PRINT CHR$(147)
1610 PRINT "ENTER COLUMN HEADING";TAB(30);
1620 GOSUB 2840
1630 PRINT CHR$(147)
1640 PRINT TAB(250);"SORT PROCEEDING
1650 D$=A$(R,C)
1660 FOR M=1 TO X
1670 IF D$=C1$(M) THEN C=M
1680 NEXT
1690 FOR J=2 TO Y
1700 R=J-1
1710 A$=A$(J,C)
1720 IF A$>=A$(R,C) THEN 1810
1730 A$(R+1,C)=A$(R,C)
1740 FOR L=1 TO X
1750 T$=A$(R,L)
1760 A$(R,L)=A$(R+1,L)
1770 A$(R+1,L)=T$
1780 NEXT L
1790 R=R-1
1800 IF R>0 THEN 1720
1810 A$(R+1,C)=A$
1820 NEXT J
1830 GOTO 110
1840 PRINT CHR$(147)
1850 PRINT "[13 SPC]TOTALS"
1860 PRINT
1870 GOSUB 3130
1880 PRINT T5(1);TAB(15)T5(2);TAB(23)T5(3)TAB(31)T5(4)
1890 PRINT "[39↑8]"
1900 PRINT:PRINT:PRINT
1910 PRINT "[12 SPC]AVERAGES"
1920 PRINT
1930 GOSUB 3130
1940 IF Y=0 THEN Y=1E-30
1950 FOR G=1 TO X
1960 T5(G)=T5(G)/Y:T5(G)=INT(100*T5(G))/100
1970 NEXT
1980 PRINT T5(1);TAB(15)T5(2);TAB(23)T5(3)TAB(31)T5(4)
1990 PRINT "[39↑8]"
2000 PRINT:PRINT:PRINT
2010 GOSUB 3080
2020 GOTO 110
2030 REM ** PREPARE TO OPEN WRITE
2040 POKE 243,122:POKE 244,2
2050 RETURN
2060 REM ** PREPARE TO CLOSE WRITE
2070 IF PEEK(625)>180 THEN 2090
2080 RETURN
2090 POKE 59411,53:T8=TI
2100 IF TI-T8<6 THEN 2100
2110 POKE 59411,61
2120 RETURN
2130 REM ** FOLLOW INPUT#1
2140 IF (ST)=0 OR (ST)=64 THEN 2170
2150 PRINT "TAPE STATUS ERROR"
2160 STOP
2170 RETURN
2180 REM ** PRINT FILE
```

```
2190 GOSUB 2900
2200 PRINT
2210 GOSUB 3080
2220 GOTO 110
2230 REM ** SELECT RECORD
2240 GOSUB 2270
2250 GOSUB 3080
2260 GOTO 110
2270 REM ** EXAM RECORD
2280 PRINT CHR$(147):PRINT:M2=0
2290 IF M2=1 THEN 2390
2300 PRINT:PRINT:PRINT
2310 PRINT "ENTER COLUMN HEADING .....";
2320 J=0:Z=0:K=0
2330 GOSUB 2840
2340 F$=A$(R,C):PRINT:M$=F$
2350 FOR M=1 TO X
2360 IF F$=C1$(M) THEN O=M:J=1
2370 NEXT
2380 IF J=0 THEN PRINT:PRINT"NO SUCH COLUMN EXISTS":PRINT:GOTO 2300
2390 PRINT:PRINT "ENTER SEARCH DATA ........";
2400 Z=0:K=0
2410 GOSUB 2840
2420 PRINT:D1=0
2430 PRINT CHR$(147):PRINT TAB(120)
2440 GOSUB 3130
2450 R$=A$(R,C)
2460 FOR R=1 TO Y
2470 Z=0
2480 FOR C=1 TO X
2490 IF A$(R,O)=R$ THEN PRINT A$(R,C);TAB(00+8*C):Z=1:R1=R:D1=1
2500 NEXT
2510 IF Z=0 THEN 2530
2520 PRINT
2530 NEXT
2540 PRINT "[39↑8]"
2550 PRINT
2560 IF D1=0 THEN PRINT "[7 SPC][REV]RECORD NOT ON FILE[OFF]"
2570 G=1
2580 RETURN
2590 GOSUB 3080
2600 GOTO 110
2610 REM ** MODIFY RECORD
2620 M2=1:O=1
2630 PRINT CHR$(147)
2640 PRINT "THE TERM SEARCH DATA BELOW REFERS TO
2650 PRINT "[9 SPC][11↑8]"
2660 PRINT "WHICH  ";C1$(1)
2670 GOSUB 2290
2680 PRINT
2690 IF D1<>0 THEN 2730
2700 PRINT
2710 GOSUB 3080
2720 GOTO 110
2730 F$=C1$(1)
2740 PRINT "[3 SPC]PROCEED TO MODIFY"
2750 PRINT
2760 FOR C=1 TO X
2770 PRINT IR$;" ";C1$(C);".......";TAB(20)
2780 GOSUB 2840
2790 L$=A$(R,C)
2800 A$(R1,C)=L$
2810 PRINT:PRINT
2820 NEXT
2830 GOTO 110
2840 REM ** CRASH PROOF INPUT
2850 OPEN 1,0
2860 INPUT#1,A$(R,C)
2870 IF A$(R,C)="" THEN 2860
2880 CLOSE 1
2890 RETURN
2900 REM ** PRINT FILE
2910 PRINT CHR$(147)
2920 GOSUB 3130
2930 S=1:V=1:T5(1)=0:T5(2)=0:T5(3)=0:T5(4)=0
2940 FOR R=1 TO Y
2950 FOR C=1 TO X
2960 T5(C)=T5(C)+VAL(A$(R,C))
2970 IF C>1 THEN 3000
2980 PRINT A$(R,C);
2990 GOTO 3010
3000 PRINT TAB(0+8*C)A$(R,C);
3010 NEXT
3020 PRINT
3030 IF S=18 THEN GOSUB 3080
3040 IF S=18 THEN S=1
3050 S=S+1
3060 NEXT
3070 RETURN
3080 REM ** PRESS KEY
3090 PRINT:PRINT "[10 SPC]PRESS ANY KEY"
3100 GET K$
3110 IF K$="" THEN 3100
3120 RETURN
3130 REM ** PRINT TABLE HEADINGS
3140 PRINT "[39↑8]"
3150 PRINT C1$(1);TAB(16)C1$(2);TAB(24)C1$(3)TAB(32)C1$(4)
3160 PRINT "[39↑8]"
3170 RETURN
```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.



We can sort the information into order, here we've arranged the stock list from the previous page into ascending order of stock code.

FILE CREATION

When an operator creates a file it is essential to guard against the inadvertant pressing of the Return key before hitting a data key. The subroutine at line 2380 prevents this on *all* INPUT statements. The option entitled 'SORT COLUMN' employs a simple bubble sort and is rather slow for long records but the extra statements for the faster merge sort would encroach on the 8K memory capacity still further. It will be appreciated that sorting records into some order is not a simple case of sorting a column. When a swop is necessary, every data item in the record must take part in the swop which complicates the process. Owners of 16K models can change the bubble to a merge sort if they find the slowness irksome.

The option 'RETRIEVE RECORD/S' allows an operator to selectively display all records which fall into required category. Thus if one of the column headings is 'AGE', we may ask of a display of all personnel of age 23. If, by chance, nobody in the file is of that age the message 'RECORD NOT ON FILE' is given.

Files are stored on a 'DATA' tape and the relevant patching routines to compensate for the 'early' ROM bugs are included in the program. It is possible to obtain further copies of a data tape by first loading in the tape by the option 'LOAD FILE' and then using 'SAVE FILE' with a blank tape. Because file creation is error prone provision exists for modifying at the end of each record and, again when the file is complete by using the option 'MODIFY FILE'.

VIC EDITOR

The problem with processing text on the VIC-20 is that the screen is rather small. No problem, just make it bigger!

The VIC-20 is a nice little computer: colour, sound and almost) high resolution graphics at a reasonable cost. Only one problem, right? The display.

It's when you try to cram a decent amount of information into a 22 by 23 screen that you wish there was some way of easily getting a larger screen. Well, there is. In fact, you can alter the configuration of the screen by simple POKEs to give any practical size up to 26 columns and 32 rows.

The screen is controlled by registers in the VIC chip. There are 16 registers which also control the colour, sound and games capabilities of the VIC.

If we call the first VIC register (at 36864) 'VIC' — a tradition begun by ones older and wiser than myself — then the registers that we are interested in are VIC, VIC + 1, VIC + 2 and VIC + 3. These registers control the screen's shape, size, location on the TV screen and the location of the screen RAM in memory.

Let's make some definitions:

VIC = 36864
HP = horizontal screen position on the TV screen (9 to 20)
VP = vertical screen position (10 to 40)
VC = number of video columns (1 to 33)
VR = number of video rows (1 to 27)

The figures in brackets are the practical limits for each value.

Choose some values and try them with:

```
POKE VIC,HP
POKE VIC+1,VP
POKE VIC+2,PEEK(VIC+2) AND 128 OR VC
POKE VIC+3,PEEK(VIC+3) AND 129 OR VR*2
```

Now try a few other values until you get the hang of what's going on.

If you experiment a little more, you may discover that:

— the cursor is totally befuddled with VC not equal to 22.

— only the first 506 screen locations can be printed to and

```
100  VIC=36864
110  VC=26:REM ** COLS
120  VR=32:REM ** ROWS
130  DEF FNA(I)=I-128*(I<128):REM ** INV CHAR FOR CURSOR
140  DEF FNB(I)=I+128*(I>127)
150  POKE VIC,9:REM ** XPOS OF SCREEN
160  POKE VIC+1,18:REM ** YPOS OF SCREEN
170  POKE VIC+2,PEEK(VIC+2) AND 128 OR VC
180  POKE VIC+3,PEEK(VIC+3) AND 129 OR VR*2
190  REM ** MOVE SCREEN DOWN BY 512 BYTES
200  POKE 36866,PEEK(36866) AND NOT 128
210  REM ** DEFINE SCREEN AND COLOUR RAM ADDRESSES
220  SC=7168:CO=37888
230  POKE 648,28:REM ** TELL VIC WHERE SCREEN IS
240  MC=SC-256:REM ** MACHINE CODE START
250  CL=MC:REM ** START OF CLEAR CODE
260  SV=MC+35:REM ** START OF SAVE CODE
270  LD=MC+66:REM ** START OF LOAD CODE
280  POKE 56,27:POKE 52,27:REM ** PROTECT SCREEN FROM
     BASIC BY LOWERING TOP OF MEMORY
290  REM ** LOAD CODE FROM DATA STATEMENTS INTO MC+I
300  FOR I=0 TO 86:READ A:POKE MC+I,A:NEXT
310  SYS MC:REM ** CLEAR SCREEN
320  P=SC:REM ** TOP LEFT
330  RV=0:REM ** REVERSE CHARACTER FLAG
340  REM ** BLINK CURSOR UNTIL A KEY IS HIT
350  POKE P,FNA(PEEK(P)):FOR I=1 TO 40
360  IF PEEK(198)<>0 THEN POKE P,FNB(PEEK(P)):GOTO 390
370  NEXT:POKE P,FNB(PEEK(P)):FOR I=1 TO 40:
     IF PEEK(198)<>0 THEN 390
380  NEXT:GOTO 350
390  A$="":GET A$
400  REM ** CLEAR SCREEN
410  IF A$="[CLS]" THEN SYS CL:P=SC:GOTO 350
420  REM ** HOME CURSOR
430  IF A$="[HOM]" THEN P=SC:GOTO 350
440  REM ** REVERSE ON
450  IF A$="[REV]" THEN RV=1:GOTO 350
460  REM ** REVERSE OFF
470  IF A$="[OFF]" THEN RV=0:GOTO 350
480  REM ** CURSOR KEYS
490  IF A$="[CD]" THEN P=P+VC
500  IF A$="[CU]" THEN P=P-VC
510  IF A$="[CR]" THEN P=P+1
520  IF A$="[CL]" THEN P=P-1
530  REM ** RETURN
540  IF A$=CHR$(13) THEN P=SC+VC+VC*INT((P-SC)/VC)
550  REM ** DELETE
560  IF A$=CHR$(20) THEN POKE P,32:P=P-1
```

only the first 512 can be POKEd to.
— 'clr' only works on the first 506 locations.

THE OUTER LIMITS

The fact that we only have 512 bytes of screen RAM available imposes a limit on the screen dimensions of 22 by 23, 26 by 19 or thereabouts. As we want a larger screen (26 by 32) we have to find the room from somewhere.

A useful source of extra RAM is the user's 3.5K. You lose a little (512 bytes) of your program space, but look what you gain. To move the screen down by 512 bytes, we clear bit 7 of VIC + 2:

```
POKE VIC+2,PEEK(VIC+2) AND NOT 128
```

We must protect our new screen by lowering the top-of-memory pointers so that BASIC can't get at it:

```
POKE 56,27:POKE 52,27
```

We can now define the screen and colour addresses for our new screen:

```
SC=7168:REM ** 7680-512
CO=37888:REM ** 38400-512
```

As the VIC's operating system doesn't know of our scheming, we have to tell it where the new screen is — you wouldn't want to miss any error messages, would you? The statement:

```
POKE 648,SC/256:REM ** 648 STORES MSB
OF SCREEN ADDRESS
```

stores the MSB of the new address.

We now have our big screen, ideal for games and any applications where we want to display a lot of information. The practical limits on the screen's dimensions depend on your TV set, but typically they are 26 columns and 32 rows.

The following program is a simple text editor in BASIC and machine code that allows you to:

— write to the screen using the cursor keys, Home, clr, Return, delete, reverse on and reverse off.
— SAVE a screenful to tape by pressing f1.
— LOAD a screenful of tape by pressing f3. (Note that as the messages PRESS PLAY ON TAPE and PRESS PLAY AND RECORD ON TAPE are suppressed to avoid messing up the screen, you will have to realise what's going on. The cursor will vanish when VIC-ED is waiting for you to press the appropriate keys on the tape recorder.)
— get a printout of the screen on a VIC printer by pressing f5.

THREE IN ONE

The program uses three machine code routines: to save, to load and to clear the screen. The 'clear' routine is necessary because VIC can only clear the first 506 bytes of the screen. Using machine code to store the screen on tape means that we can save it in binary which is about 20 times faster than a normal data save.

The machine code is loaded into a 256 byte block of RAM just below the screen address by reading values from the data statements at the end of the program. The entry points defined in the program for the routines are: CL for 'clear', SV for 'save' and LD for 'load'. ▶

```
570  IF P<SC THEN P=P+(26*32)
580  IF P>=SC+(26*32) THEN P=P-(26*32)
590  IF A$="[CD]" OR A$="[CU]" OR A$="[CR]" OR A$="[CL]"
     THEN 350
600  IF A$=CHR$(20) OR A$=CHR$(13) THEN 350
610  IF A$="[F1]" THEN GOSUB 840:GOTO 350:REM ** F1
     SAVES TO TAPE
620  IF A$="[F3]" THEN GOSUB 850:REM ** F3 LOADS FROM
     TAPE
630  IF A$="[F5]" THEN GOSUB 730:GOTO 350:REM ** F5
     PRINTS SCREEN
640  REM ** GET SCREEN CODE FROM ASCII INPUT
650  A=ASC(A$):IF A>31 AND A<64 THEN GOTO 700
660  IF A>63 AND A<96 OR A>160 AND A<192 THEN A=A-64:
     GOTO 700
670  IF A=255 THEN A=94:GOTO 700
680  IF A>191 AND A<224 THEN A=A-128:GOTO 700
690  GOTO 350:REM ** UNPRINTABLE
700  IF RV=1 THEN A=FNA(A):REM ** REVERSE FLAG ON
710  POKE P,A:IF P<SC+VR*VC-1 THEN P=P+1
720  GOTO 350:REM ** NEXT CHARACTER
730  REM ** SCREEN COPY
740  A$=CHR$(145):IF (PEEK(36869) AND 2)=2 THEN
     A$=CHR$(17)
750  OPEN 4,4:IF A$=CHR$(17) THEN CLOSE 4:OPEN 4,4,7
760  PRINT#4:A=SC-VC:FOR B=0 TO VR-1:B$=A$:A=A+VC:
     FOR C=A TO A+VC-1
770  D=PEEK(C):IF D>128 THEN D=D-128:RV=1:B$=B$+CHR$(18)
780  IF D<32 THEN D=D+64:GOTO 820
790  IF (D>31) AND (D<64) THEN 820
800  IF (D>63) AND (D<96) THEN D=D+32:GOTO 820
810  IF (D>95) AND (D<128) THEN D=D+64
820  B$=B$+CHR$(D):IF RV=1 THEN B$=B$+CHR$(146):RV=0
830  NEXT:PRINT#4,B$:NEXT:PRINT#4:CLOSE 4:RETURN
840  POKE 648,30:PRINT "[HOM][17 CD]":SYS SV:
     POKE 648,28:RETURN
850  POKE 648,30:PRINT "[HOM][15 CD]":SYS LD:
     POKE 648,28:RETURN
860  REM ** DATA FOR MACHINE CODE ROUTINES
870  REM ** STARTING AT CL,SV AND LD
880  DATA 162,0,169,32,157,0,28,157,0,29
890  DATA 157,0,30,157,0,31,169,0,157,0
900  DATA 148,157,0,149,157,0,150,157,0
910  DATA 151,232,208,225,96,234,162,1
920  DATA 160,255,32,186,255
930  DATA 169,0,32,189,255,169,28,133,2,169,0,133,1
940  DATA 169,1,162,64,160,31,32,216,255,96,234
950  DATA 162,1,160,255,32,186,255,169,0,32,189
960  DATA 255,162,255,160,255,32,213,255,96,234
```

Listing 1. The program for increasing the size of the VIC-20 screen.

Thus, the statement SYS CL will clear the screen.

When you have typed the program, first check very carefully for mistakes in your copying, then SAVE it *before* running — machine code has a nasty habit of killing itself.

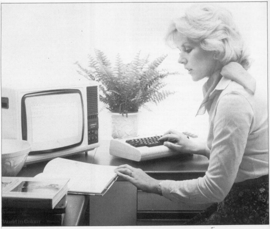
When it runs correctly, you will find that for a few moments the screen is full of rubbish — this should vanish as soon as the program loads the machine code and clears the screen. If there are any glitches at this point, look for errors in the machine code.

The program will run on a 3.5K or 6.5K VIC with any practical screen sizes and will fit into the basic VIC if you remove some of the REMs. It *won't* run on a 16K machine. It's all to do with the shifting screen and RAM conflicting — blame Commodore, not me! My advice is to remove the expansion RAM pack and 'downgrade' your VIC.

References: **The VIC Programmers Reference Manual** — Commodore and **VIC Revealed** — Nick Hampshire

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

```
** VIC-ED BY P.HINTJENS **

VIC-ED IS A SCREEN EDITOR
FOR THE COMMODORE VIC-20 -
IT GIVES THE USER A 26*32
SCREEN AND VARIOUS CURSOR
FUNCTIONS.

VIC-ED ALLOWS THE USER TO
SAVE SCREENS OF TEXT TO A
TAPE IN BINARY. AN AVERAGE
SAVE TAKES ABOUT 40 SECS.

THE USER CAN ALSO MAKE A
HARD COPY OF THE SCREEN IF
A VICPRINTER IS AROUND.

THE FUNCTION KEYS USED ARE

 F1  SAVE-TO-TAPE
 F3  LOAD-FROM-TAPE
 F5  PRINT THE SCREEN

▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼

THE PROGRAM DEMONSTRATES
WHAT CAN BE DONE AND ISN'T
AN END IN ITSELF.  IT WILL
LEND ITSELF TO SOME USEFUL
EXPANSION.
```



Subscriptions

Personally, we think you'll like our approach to microcomputing. Each month, we invite our readers to join us in an abundance of feature articles, projects, general topics, software listings, news and reviews — all to help committed micro users make more of their microcomputers.

However, if you've ever missed a copy of Computing Today on the newstands, you'll not need us to tell you how valuable a subscription can be. Subscribe to CT and for a whole year you can sit back, assured that each issue, lovingly wrapped, will find its way through your letter box.

And it's not difficult! All you have to do is fill in the form below, cut it out and send it (or a photocopy) with your cheque or Postal Order (made payable to ASP Ltd) to:

COMPUTING TODAY Subscriptions,
513 London Road,
Thornton Heath,
Surrey CR4 6AR.

Alternatively, you can pay by Access or Barclaycard in which case, simply fill in your card number, sign the form and send it off. Please don't send in your card.

Looking for a magazine with a professional approach with material written by micro users for micro users? Why not do yourself a favour and make 1983 the year you subscribe to Computing Today and we'll give you a truly personal approach to microcomputing.

SUBSCRIPTION ORDER FORM

Cut out and SEND TO :
COMPUTING TODAY Subscriptions
513 LONDON ROAD,
THORNTON HEATH,
SURREY CR4 6AR.

Please commence my subscription to Computing Today with the .... issue.

SUBSCRIPTION RATES (tick ☐ as appropriate)

£12.70 for 12 issues UK ☐
£16.35 for 12 issues Overseas Surface ☐
£36.00 for 12 issues Overseas Air Mail ☐

I am enclosing my (delete as necessary) cheque/ Postal Order/ International Money Order for £..........
(made payable to ASP Ltd)
or
Debit my Access/ Barclaycard*
(*delete as necessary)

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

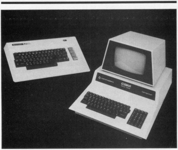
Please use BLOCK CAPITALS and include postcodes.

NAME (Mr/ Mrs/ Miss) ....
delete accordingly
ADDRESS ....
....
.... POSTCODE ....
Signature ....
Date ....

Roger Beresford

COMMODORE COMMUNICATIONS

If you thought that loading software from disc was fast you've never downloaded from another machine!



This article describes a machine code utility to let two PETs, a PET and VIC or two VICs exchange programs via a link between the user ports. This link involves no additional circuitry, merely a 10 core screened cable with the appropriate connector on each end (see Fig. 1).

The general procedure would be to load the machine code, via a BASIC loader, into both machines. Programs may then be exchanged at speeds in excess of disc loads by giving a SYS command on each machine:

```
On a VIC + 3K  SYS 7424,S  to send
               SYS 7424,R  to receive

On an 8K PET   SYS 7936,S  to send
               SYS 7936,R  to receive
```

As the machine code is loaded into high memory in each case, it needs protecting. If loaded in Hex using the monitor, it contains code to protect itself, but if loaded from a BASIC loader it will need protecting from BASIC prior to running with:

```
PET as above    POKE 52,0:POKE 53,30
                POKE 48,0:POKE 49,30
                CLR

VIC as above    POKE 51,0:POKE 52,29
                POKE 55,0:POKE 56,29
                CLR
```

It is vital to start the receiver first, both out of consideration for the hardware, as both user ports could be configured as outputs, and for the need to catch the first byte of the transfer.

THE IDEA

The idea and requirement for these routines arose after the acquisition of a VIC computer without a cassette player. Therefore, this computer had to share a cassette player with a PET system and it always seemed that the cassette was on the wrong machine. The power should be turned off before plugging in or unplugging the cassette, or anything else for that matter, as the VIA lines are unbuffered and the VIA chip is easily blown by accidental shorts. Turning the power off also loses the program. Using the Communicator program loaded into the VIC from the cassette and into the PET from disc, it is possible to exchange programs at will, very quickly. This is not network software but is a little way towards this very popular subject.

Listing 1 gives both the source and object code for the PET version of the program. As it is symbolic, it may be easily relocated by readers with an assembler. The program calls three ROM routines that would have to be changed for other than New ROMs (3.0); what I believe to be BASIC 4.0 versions are given in the listing. The VIC version is structurally identical, though the user port is mapped to a different address and is the 'B' side of the VIA, whereas in the PET the 'A' side and CB2 are used. This entails some changes to the handshaking of the data across the link. Listing 2 gives the VIC version assembled for a VIC with a 3K memory cartridge, though this again is easily relocated for other VIC configurations. Listing 3 gives a BASIC loader version for the VIC described above and Listing 4 gives the BASIC loader for a 4.0 ROM 8K PET.

HOW THE PROGRAM WORKS

The two machine code programs might look fairly fearsome but they are structured for simplicity. The main sections will be discussed in more detail below.

PROTECT This section alters the top of memory pointers. This could be done from BASIC to save a few bytes if memory is short.

START This section looks beyond the SYS for the 'S' or 'R' signifying Send or Receive. If neither are present it will jump to the BASIC Syntax Error routine. Depending on whether an 'S' or an 'R' follows the SYS, the VIA data lines are set up for input or output.

SETUP This section first moves the CHRGET pointer past the 'S' or 'R' and then sets up the ACR (Auxilliary Control Register) and PCR (Peripheral Control Register) with the correct bit pattern for using CB2 as Data Valid/Busy and CA1 (CB1 on VIC) as data strobes.

SETBASPOINT Takes a copy of the 'Start of BASIC' vector and saves it at BASPOINT. It then checks the DDRA (Data Direction Register) and branches to either LISTEN or TRANSMIT.

LISTEN The first part signals to the other CBM machines that it is ready to receive data. The next subroutine waits for the data strobe from the other CBM and then loads the data from the VIA into the BASIC text area pointed to by BASPOINT. Next, a check is made to see if the end of the program has been received. (The end of a program is marked by three consecutive zeroes.) If the end is found, the program jumps to the ENDFOUND routine. If it is not the end, the program loops back to the beginning of the LISTEN routine.

TRANSMIT This is similar to LISTEN but in reverse. It waits for the receiver to signal ready and then loads the data from the BASIC text area into the VIA. A strobe then indicates to the receiver that the data is valid. A similar endcheck to LISTEN is performed and if it has not reached the end, it loops back for the next character. On finding the end the program branches to SENDEND.

ENDFOUND This section realigns the pointers so that BASIC knows the length of the transmitted program. A similar jump to CLR completes the LISTEN routine.

The rest of the program consists of the various subroutines called by the main program above.

USREADY Sends a pulse on the CB2 line to the CA1/CB1 input of the other machine, to signal Busy Clear or Data Valid.

THEMREADY Waits for a pulse on CA1/CB1 as above.

INCTXT Increments the CHRGET pointer and loads the next character from the BASIC text.

INCPTR Increments the BASPOINT pointer.

ENDCHECK This routine looks back over the received characters to check for three zeroes and sets the end flag at $00 accordingly.

DECPTR Decrements the pointer used by ENDCHECK to look back through the BASIC text.

CHECKEND Similar to ENDCHECK but the routine looks forward to trap the end condition before it is transmitted.

SENDEND Transmits three zeroes.

TESTING THE PROGRAM

When using machine code, all the protection built into BASIC is lost and it is easy to set the computer into a condition where it is stuck in a loop, 'crashed' in the vernacular. In order to be able to interrupt the program

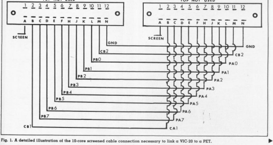


Fig. 1. A detailed illustration of the 10-core screened cable connection necessary to link a VIC-20 to a PET. ▶

with the 'Stop' key, there needs to be a call in the main loop to either $F30F, which checks for the Stop key and then performs a stop, or to $F301 which tests the Stop key and sets the zero flag. (This enables any tidying up to be done before a program exit.) However, in order to use these routines, the interrupts must be enabled which does slow up the data transmission considerably. In the final version given here, there is no provision for using the Stop key, so errors will probably cause a crash and necessitate turning both computers off. It is, therefore, important to save a copy of the program ***before*** testing.

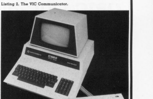
As two machines and two programs are involved there is at least twice the scope for error. The most likely source of error, apart from copying errors, is having the interrupt flags set incorrectly before starting. The RECEIVE routine clears the IFR before starting. The RECEIVE routine clears the IFR before starting, though as a result of correcting a mistake halfway through a transmission, it is possible to get locked in a loop with each computer waiting for the other to send 'Ready'. Taking care in running RECEIVE first should help here.

```
033A                 ;*****************************************
033A                 ;*
033A                 ;* USER PORT COMMUNICATOR
033A                 ;*
033A                 ;* (RUN RECEIVER FIRST)
033A                 ;*
033A                 ;* PET PROGRAM
033A                 ;*
033A                 ;*****************************************
033A                 ;*
033A                 ;* SYS 7936,S ...... TO SEND
033A                 ;*
033A                 ;* SYS 7936,R ...... TO RECEIVE
033A                 ;*
033A                 ;*****************************************
033A                 ;*
033A                 ;* VARIABLES
033A                 ;*
033A                 ;*****************************************
EB40                 VIAORG=$EB40 LOCATION OF VIA IN MEMORY
EB40                 BREG=VIAORG
EB41                 HAND=VIAORG+1
EB42                 DDRB=VIAORG+2
EB43                 DDRA=VIAORG+$03
EB4B                 ACR=VIAORG+$0B
EB4C                 PCR=VIAORG+$0C
EB4D                 IFR=VIAORG+$0D
EB4F                 AREG=VIAORG+$0F
C579                 CLR=$C579 BASIC CLR ROUTINE .......... ($B5EC)
C439                 REBUILD=$C439 REBUILD BASIC POINTERS . ($B4B6)
CE03                 SYNERR=$CE03 SYNTAX ERROR ............ ($BF00)
0001                 BASPOINT=$01 POINTER FOR BASIC TEXT
0028                 BASSTART=$28 START OF BASIC POINTER
0034                 MEMTOP=$34 TOP OF MEMORY POINTER
002A                 BASEND=$2A  END  OF          BASIC POINTER
0077                 TXTPTR=$77 CHRGET POINTER
0088                 RNDSEED=$88 TEMPORARY STORAGE OF A POINTER
033A                 ;
033A                 *=$1F00
1F00                 ;
1F00                 ;*******************************************
1F00                 ;*
1F00                 ;* MAIN PROGRAM
1F00                 ;*
1F00                 ;*******************************************
1F00                 ;
1F00 A200            PROTECT     LDX #<PROTECT      PROTECT M/C
1F02 A91F                        LDA #>PROTECT
1F04 8535                        STA MEMTOP+1
1F06 8634                        STX MEMTOP
1F08 20A81F          START       JSR INCTXT
1F0B C952                        CMP #'R            IS IT RECEIVE
1F0D F007                        BEQ RECEIVE
1F0F C953                        CMP #'S            IS IT SEND
1F11 F00B                        BEQ SEND
1F13 4C03CE                      JMP SYNERR
1F16 A900            RECEIVE     LDA #$00           SET PA0-7 AS I/P
1F18 8D43E8                      STA DDRA
1F1B 4C231F                      JMP SETUP
1F1E A9FF            SEND        LDA #$FF           SET PA0-7 AS O/P
1F20 8D43E8                      STA DDRA
1F23 20A81F          SETUP       JSR INCTXT         MOVE POINTER PAST S/R
1F26 78                          SEI
1F27 AD4BE8                      LDA ACR
1F2A 0901                        ORA #$01           ENABLE LATCHING
1F2C 29E3                        AND #$E3           DISABLE SHIFT REGISTER
1F2E 8D4BE8                      STA ACR
1F31 AD4CE8                      LDA PCR
1F34 09C1                        ORA #$C1           +VE TRANSITION ON CA1
1F36 29DF                        AND #$DF           SET CB2 LOW
1F38 8D4CE8                      STA PCR
1F3B A528            SETBASPOINT LDA BASSTART
1F3D 8501                        STA BASPOINT
1F3F A529                        LDA BASSTART+1
1F41 8502                        STA BASPOINT+1
1F43 AD43E8                      LDA DDRA
1F46 D01D                        BNE TRANSMIT
1F48 AD41E8                      LDA HAND           RESET IFR BEFORE START
1F4B 20921F          LISTEN      JSR USREADY        PUT BUSY CLEAR
1F4E 20A01F                      JSR THEMREADY      STROBE CHECK
1F51 A000                        LDY #$00
1F53 AD41E8                      LDA HAND           GET DATA FROM USER PORT
1F56 9101                        STA (BASPOINT),Y   PUT IN BASIC TEXT
1F58 20BA1F                      JSR ENDCHECK
1F5B A500                        LDA $00            LOAD END FLAG
1F5D D024                        BNE ENDFOUND
1F5F 20B31F                      JSR INCPTR
1F62 4C4B1F                      JMP LISTEN
1F65 20A01F          TRANSMIT    JSR THEMREADY      CHECK FOR BUSY CLEAR
1F68 A000                        LDY #$00
1F6A B101                        LDA (BASPOINT),Y   GET DATA FROM BASIC TEXT
1F6C 8D41E8                      STA HAND           PUT DATA TO USER PORT
1F6F 20921F                      JSR USREADY        STROBE THE CB2 LINE
1F72 20B31F                      JSR INCPTR
1F75 20E11F                      JSR CHECKEND
1F78 A500                        LDA $00            LOAD END FLAG
1F7A F0E9                        BEQ TRANSMIT
1F7C 20EF1F                      JSR SENDEND
1F7F 58                          CLI
1F80 4C79C5                      JMP CLR
1F83 A501            ENDFOUND    LDA BASPOINT       REALIGN BASIC POINTERS
1F85 852A                        STA BASEND
1F87 A502                        LDA BASPOINT+1
1F89 852B                        STA BASEND+1
1F8B 58                          CLI
1F8C 2039C4                      JSR REBUILD        THE LINK POINTERS
1F8F 4C79C5                      JMP CLR
1F92                 ;***********************************
1F92                 ;*
1F92                 ;* SUBROUTINES
1F92                 ;*
1F92                 ;***********************************
1F92                 ;
1F92 AD4CE8          USREADY     LDA PCR
1F95 09E0                        ORA #$E0           CB2 HIGH
1F97 8D4CE8                      STA PCR
1F9A 29DF                        AND #$DF           CB2 LOW AGAIN
1F9C 8D4CE8                      STA PCR
1F9F 60                          RTS
1FA0                 ;
1FA0 AD4DE8          THEMREADY   LDA IFR            BUSY OR STROBE
1FA3 2902                        AND #$02
1FA5 F0F9                        BEQ THEMREADY
1FA7 60                          RTS
1FA8                 ;
1FA8 A000            INCTXT      LDY #$00           INCREMENT CHRGET POINTER
1FAA E677                        INC TXTPTR
1FAC D002                        BNE NOCARRY
1FAE E678                        INC TXTPTR+1
1FB0 B177            NOCARRY     LDA (TXTPTR),Y
1FB2 60                          RTS
1FB3                 ;
1FB3 E601            INCPTR      INC BASPOINT       INCREMENT BASIC POINTER
1FB5 D002                        BNE NOTCARRY
1FB7 E602                        INC BASPOINT+1
1FB9 60              NOTCARRY    RTS
1FBA                 ;
1FBA A501            ENDCHECK    LDA BASPOINT       END OF PROGRAM
1FBC 8588                        STA RNDSEED
1FBE A502                        LDA BASPOINT+1
1FC0 8589                        STA RNDSEED+1
1FC2 A203                        LDX #$03
1FC4 A000                        LDY #$00
1FC6 B188            LOOPA       LDA (RNDSEED),Y
1FC8 D00B                        BNE NOTEND
1FCA 20DA1F                      JSR DECPTR
1FCD CA                          DEX
1FCE D0F6                        BNE LOOPA
1FD0 A9FF            FOUNDEND    LDA #$FF
1FD2 8500                        STA $00            SET END FLAG
1FD4 60                          RTS
1FD5 A900            NOTEND      LDA #$00           CLEAR END FLAG
1FD7 8500                        STA $00
1FD9 60                          RTS
1FDA                 ;
1FDA C688            DECPTR      DEC RNDSEED        DECREMENT BASIC POINTER
1FDC D002                        BNE NOBORROW
1FDE C689                        DEC RNDSEED+1
1FE0 60              NOBORROW    RTS
1FE1                 ;
1FE1 A000            CHECKEND    LDY #$00
1FE3 B101            LOOPB       LDA (BASPOINT),Y
1FE5 D0EE                        BNE NOTEND
1FE7 C8                          INY
1FE8 C003                        CPY #$03
1FEA 90F7                        BCC LOOPB
1FEC 4CD01F                      JMP FOUNDEND
1FEF                 ;
1FEF A203            SENDEND     LDX #$03           SEND THREE ZEROS
1FF1 20A01F          LOOPC       JSR THEMREADY
1FF4 A900                        LDA #$00
1FF6 8D41E8                      STA HAND
1FF9 20921F                      JSR USREADY
1FFC CA                          DEX
1FFD D0F2                        BNE LOOPC
1FFF 60                          RTS
```

Listing 1. The PET Communicator.

```
033A                   ;****************************************
033A                   ;*
033A                   ;* USER PORT COMMUNICATOR
033A                   ;*
033A                   ;* (RUN RECEIVER FIRST)
033A                   ;*
033A                   ;* VIC PROGRAM
033A                   ;*
033A                   ;***************************************
033A                   ;*
033A                   ;* SYS 7424,S ....... TO SEND
033A                   ;*
033A                   ;* SYS 7424,R ....... TO RECEIVE
033A                   ;*
033A                   ;***************************************
033A                   ;*
033A                   ;* VARIABLES
033A                   ;*
033A                   ;***************************************
9110                   VIAORG=$9110 LOCATION OF VIA IN MEMORY
9110                   BREG=VIAORG
9111                   HAND=VIAORG+1
9112                   DDRB=VIAORG+2
9113                   DDRA=VIAORG+$03
911B                   ACR=VIAORG+$0B
911C                   PCR=VIAORG+$0C
911D                   IFR=VIAORG+$0D
911F                   AREG=VIAORG+$0F
C660                   CLR=$C660 BASIC CLR ROUTINE
C533                   REBUILD=$C533 REBUILD BASIC POINTERS
CF08                   SYNERR=$CF08 SYNTAX ERROR
0001                   BASPOINT=$01
002B                   BASSTART=$2B
0037                   MEMTOP=$37 TOP OF MEMORY POINTER
002D                   BASEND=$2D
007A                   TXTPTR=$7A CHRGET POINTER
008B                   RNDSEED=$8B TEMPORARY POINTER STORAGE
033A                   ;
033A                   *=$1D00
1D00                   ;
1D00                   ;*******************************************
1D00                   ;*
1D00                   ;* MAIN PROGRAM
1D00                   ;*
1D00                   ;*******************************************
1D00                   ;
1D00  A200     PROTECT     LDX  #<PROTECT      PROTECT M/C
1D02  A91D                 LDA  #>PROTECT
1D04  8538                 STA  MEMTOP+1
1D06  8637                 STX  MEMTOP
1D08  20A81D   START       JSR  INCTXT
1D0B  C952                 CMP  #'R            IS IT RECEIVE
1D0D  F007                 BEQ  RECEIVE
1D0F  C953                 CMP  #'S            IS IT SEND
1D11  F00B                 BEQ  SEND
1D13  4C08CF               JMP  SYNERR
1D16  A900     RECEIVE     LDA  #$00           SET PB0-7 AS I/P
1D18  8D1291               STA  DDRB
1D1B  4C231D               JMP  SETUP
1D1E  A9FF     SEND        LDA  #$FF           SET PB0-7 AS O/P
1D20  8D1291               STA  DDRB
1D23  20A81D   SETUP       JSR  INCTXT         MOVE POINTER PAST S/R
1D26  78                   SEI
1D27  AD1B91               LDA  ACR
1D2A  0902                 ORA  #$02           ENABLE LATCHING
1D2C  29E3                 AND  #$E3           DISABLE SHIFT REGISTER
1D2E  8D1B91               STA  ACR
1D31  AD1C91               LDA  PCR
1D34  09D0                 ORA  #$D0           +VE TRANSITION ON CB1
1D36  29DF                 AND  #$DF           SET CB2 LOW
1D38  8D1C91               STA  PCR
1D3B  A52B     SETBASPOINT LDA  BASSTART
1D3D  8501                 STA  BASPOINT
1D3F  A52C                 LDA  BASSTART+1
1D41  8502                 STA  BASPOINT+1
1D43  AD1291               LDA  DDRB
1D46  D01D                 BNE  TRANSMIT
1D48  AD1091               LDA  BREG           RESET IFR BEFORE START
1D4B  20921D   LISTEN      JSR  USREADY        PUT BUSY CLEAR
1D4E  20A01D               JSR  THEMREADY      STROBE CHECK
1D51  A000                 LDY  #$00
1D53  AD1091               LDA  BREG           GET DATA FROM USER PORT
1D56  9101                 STA  (BASPOINT),Y   PUT IN BASIC TEXT
1D58  20BA1D               JSR  ENDCHECK
1D5B  A500                 LDA  $00            LOAD END FLAG
1D5D  D024                 BNE  ENDFOUND
1D5F  20B31D               JSR  INCPTR
1D62  4C4B1D               JMP  LISTEN
1D65  20A01D   TRANSMIT    JSR  THEMREADY      CHECK FOR BUSY CLEAR
1D68  A000                 LDY  #$00
1D6A  B101                 LDA  (BASPOINT),Y   GET DATA FROM BASIC TEXT
1D6C  8D1091               STA  BREG           PUT DATA TO USER PORT
1D6F  20921D               JSR  USREADY        STROBE THE CB2 LINE
1D72  20B31D               JSR  INCPTR
1D75  20E11D               JSR  CHECKEND
1D78  A500                 LDA  $00            LOAD END FLAG
1D7A  F0E9                 BEQ  TRANSMIT
1D7C  20EF1D               JSR  SENDEND
1D7F  58                   CLI
1D80  4C60C6               JMP  CLR
1D83  A501     ENDFOUND    LDA  BASPOINT       REALIGN BASIC POINTERS
1D85  852D                 STA  BASEND
1D87  A502                 LDA  BASPOINT+1
1D89  852E                 STA  BASEND+1
1D8B  58                   CLI
1D8C  2033C5               JSR  REBUILD        THE LINK POINTERS
1D8F  4C60C6               JMP  CLR
1D92                   ;*********************************
1D92                   ;*
1D92                   ;* SUBROUTINES
1D92                   ;*
1D92                   ;*********************************
1D92                   ;
1D92  AD1C91   USREADY     LDA  PCR
1D95  09E0                 ORA  #$E0           CB2 HIGH
1D97  8D1C91               STA  PCR
1D9A  29DF                 AND  #$DF           CB2 LOW AGAIN
1D9C  8D1C91               STA  PCR
1D9F  60                   RTS
1DA0                   ;
1DA0  AD1D91   THEMREADY   LDA  IFR            BUSY OR STROBE
1DA3  2910                 AND  #$10
1DA5  F0F9                 BEQ  THEMREADY
1DA7  60                   RTS
1DA8                   ;
1DA8  A000     INCTXT      LDY  #$00           INCREMENT CHRGET POINTER
1DAA  E67A                 INC  TXTPTR
1DAC  D002                 BNE  NOCARRY
1DAE  E67B                 INC  TXTPTR+1
1DB0  B17A     NOCARRY     LDA  (TXTPTR),Y
1DB2  60                   RTS
1DB3                   ;
1DB3  E601     INCPTR      INC  BASPOINT       INCREMENT BASIC POINTER
1DB5  D002                 BNE  NOTCARRY
1DB7  E602                 INC  BASPOINT+1
1DB9  60       NOTCARRY    RTS
1DBA                   ;
1DBA  A501     ENDCHECK    LDA  BASPOINT
1DBC  858B                 STA  RNDSEED
1DBE  A502                 LDA  BASPOINT+1
1DC0  858C                 STA  RNDSEED+1
1DC2  A203                 LDX  #$03
1DC4  A000                 LDY  #$00
1DC6  B18B     LOOPA       LDA  (RNDSEED),Y
1DC8  D00B                 BNE  NOTEND
1DCA  20DA1D               JSR  DECPTR
1DCD  CA                   DEX
1DCE  D0F6                 BNE  LOOPA
1DD0  A9FF     FOUNDEND    LDA  #$FF
1DD2  8500                 STA  $00            SET END FLAG
1DD4  60                   RTS
1DD5  A900     NOTEND      LDA  #$00           CLEAR END FLAG
1DD7  8500                 STA  $00
1DD9  60                   RTS
1DDA                   ;
1DDA  C68B     DECPTR      DEC  RNDSEED        DECREMENT BASIC POINTER
1DDC  D002                 BNE  NOBORROW
1DDE  C68C                 DEC  RNDSEED+1
1DE0  60       NOBORROW    RTS
1DE1                   ;
1DE1  A000     CHECKEND    LDY  #$00
1DE3  B101     LOOPB       LDA  (BASPOINT),Y
1DE5  D0EE                 BNE  NOTEND
1DE7  C8                   INY
1DE8  C003                 CPY  #$03
1DEA  90F7                 BCC  LOOPB
1DEC  4CD01D               JMP  FOUNDEND
1DEF                   ;
1DEF  A203     SENDEND     LDX  #$03           SEND THREE ZEROS
1DF1  20A01D   LOOPC       JSR  THEMREADY
1DF4  A900                 LDA  #$00
1DF6  8D1091               STA  BREG
1DF9  20921D               JSR  USREADY
1DFC  CA                   DEX
1DFD  D0F2                 BNE  LOOPC
1DFF  60                   RTS
```

Listing 2. The VIC Communicator.



NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

```
100 READ L,H
110 FOR I=L TO H
120 READ DT
130 POKE I,DT
140 NEXT
150 PRINT CHR#(147)
1000 DATA 7424, 7679
1010 DATA 162,0,169,29,133,56,134,55,32,168,29,201,82,240,7,201
1020 DATA 83,240,11,76,8,207,169,0,141,18,145,76,35,29,169,255
1030 DATA 141,18,145,32,168,29,120,173,27,145,9,2,41,227,141,27
1040 DATA 145,173,28,145,9,208,41,223,141,28,145,165,43,133,1,165
1050 DATA 44,133,2,173,18,145,208,29,173,16,145,32,146,29,32,160
1060 DATA 29,160,0,173,16,145,145,1,32,186,29,165,0,208,36,32
1070 DATA 179,29,76,75,29,32,160,29,160,0,177,1,141,16,145,32
1080 DATA 146,29,32,179,29,32,225,29,165,0,240,233,32,239,29,88
1090 DATA 76,96,198,165,1,133,45,165,2,133,46,88,32,51,197,76
1100 DATA 96,198,173,28,145,9,224,141,28,145,41,223,141,28,145,96
1110 DATA 173,29,145,41,16,240,249,96,160,0,230,122,208,2,230,123
1120 DATA 177,122,96,230,1,208,2,230,2,96,165,1,133,139,165,2
1130 DATA 133,140,162,3,160,0,177,139,208,11,32,218,29,202,208,246
1140 DATA 169,255,133,0,96,169,0,133,0,96,198,139,208,2,198,140
1150 DATA 96,160,0,177,1,208,238,200,192,3,144,247,76,208,29,162
1160 DATA 3,32,160,29,169,0,141,16,145,32,146,29,202,208,242,96
1170 DATA 10
```

Listing 3. BASIC loader for the VIC-20.

```
100 READ L,H
110 FOR I=L TO H
120 READ DT
130 POKE I,DT
140 NEXT
1000 DATA 7936, 8191
1010 DATA 162,0,169,31,133,53,134,52,32,168,31,201,82,240,7,201
1020 DATA 83,240,11,76,0,191,169,0,141,67,232,76,35,31,169,255
1030 DATA 141,67,232,32,168,31,120,173,75,232,9,1,41,227,141,75
1040 DATA 232,173,76,232,9,193,41,223,141,76,232,165,40,133,1,165
1050 DATA 41,133,2,173,67,232,208,29,173,65,232,32,146,31,32,160
1060 DATA 31,160,0,173,65,232,145,1,32,186,31,165,0,208,36,32
1070 DATA 179,31,76,75,31,32,160,31,160,0,177,1,141,65,232,32
1080 DATA 146,31,32,179,31,32,225,31,165,0,240,233,32,239,31,88
1090 DATA 76,236,181,165,1,133,42,165,2,133,43,88,32,182,180,76
1100 DATA 236,181,173,76,232,9,224,141,76,232,41,223,141,76,232,96
1110 DATA 173,77,232,41,2,240,249,96,160,0,230,119,208,2,230,120
1120 DATA 177,119,96,230,1,208,2,230,2,96,165,1,133,136,165,2
1130 DATA 133,137,162,3,160,0,177,136,208,11,32,218,31,202,208,246
1140 DATA 169,255,133,0,96,169,0,133,0,96,198,136,208,2,198,137
1150 DATA 96,160,0,177,1,208,238,200,192,3,144,247,76,208,31,162
1160 DATA 3,32,160,31,169,0,141,65,232,32,146,31,202,208,242,96
```

Listing 4. BASIC loader for a CBM 2000/3000/4000.

ORIC-1,VIC-20

The meanest game of space invaders you'll ever play!

FEATURES

- Three invader types
- Written in machine code
- Mystery scoring mother ships
- Bonus bases every 2000 points
- High score register
- Full sound effects and colour

You must defend earth from the hordes of bomb dropping space invaders, using your earth bound laser base. The more invaders you hit, the faster they come - leading to a gun-blazing shoot-out as finale.

A great shoot-'em-up, all action arcade game, for the 16K or 48K ORIC-1 or unexpanded VIC-20.

Send cheque or P.O. for £4.00 (postage paid) stating computer type and memory size to:-

ARCADIA
SOFTWARE
FREEPOST, SWANSEA, SA3 4ZZ

THE INSTITUTION OF ANALYSTS & PROGRAMMERS

An association which endorses the status of its members, encourages their high standards, assists their careers and promotes their interests is the essential foundation of every profession.

The Institution of Analysts & Programmers is the leading association for those engaged in systems analysis or computer programming for Commerce, Industry or Public Service. Membership of the Institution, as shown by the designatory letters Cmpn.I.A.P., F.I.A.P., M.I.A.P. and A.M.I.A.P. is widely recognised and respected. The Institution is the supervising authority for the Copyright Register whose protective legal service is available to all (members and non-members) who write original programs.

If your computer practice could make you eligible to join the Institution or if you wish to secure your right to royalties through the Copyright Register write or telephone

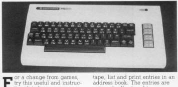
01 - 898 2385

The General Secretary
The Institution Of Analysts & Programmers
WYE HOUSE, TANGIER ROAD, RICHMOND, TW10 5DW

Pieter Hintjens

ADDRESS BOOK

Find out how to address the problem of creating a useful and instructive data base program.



For a change from games, try this useful and instructive data base program on your VIC-20. Using the excellent screen editor, you can enter, alter, delete, store on tape, list and print entries in an address book. The entries are automatically stored in alphabetical order by name and each entry can be up to 88 characters long (four screen lines).

To adapt the size of the address book to your memory size, alter the variable MAX at the start of the program. On the unexpanded VIC-20, set MAX = 25; with a 3K expansion, set MAX = 50 and with a 16K expansion set MAX = 250. If you are using entries that are all quite short, you can increase these limits accordingly.

When you run the address book program, you will appear to be in normal operating mode; the cursor will flash and all the editing keys will work. However, the program is in control and only the following commands will be understood:

/entry	The '/' tells the program to enter the entry into the address book. The entry must be in the format: /J.A.SMITH ; . . . where the name has exactly four initials and is followed by a space. If the name only has two initials, spaces should be entered. The entry is ordered as SMITHJ.A. . . . ie by surname and then initials. The data you put after the name is not used in the ordering.
@entry	Using the same principles of surname and initials for ordering, the program will search for the name and address entered and substitute the new data for the data that was previously attached to the name.
*entry	Again using the same ordering principles, this command will delete the entry from the address book.
END	Exits the program and resets the normal operation of the Stop key.
HELP	Lists these commands on the screen.
LIST (start)	Lists the address book entries starting at the first or (start). After each entry is listed, the program waits for you to press a key; if you press Stop the list will stop, otherwise the next entry will be listed. (start) can be any number of significant letters and the same surname-initials order applies. Eg LIST SM will list from the first surname starting with 'SM'.
LOAD (name)	Loads the address book called (name) off tape.
SAVE (name)	Saves the address book to tape and calls it (name).
MERGE (name)	Finds the address book called (name) on tape and merges the entries in it with the address book in memory.
PRINT	Prints the address book on the printer. The format of the printed entries depends on the special control characters used. (See later.)
FREE	This command will tell you how many entries unused there are and the amount of free memory available.

In all the above commands, the parts in brackets are optional. When giving a name for a LOAD, SAVE or MERGE, no quote marks are necessary. To use a command, simply type in the command and press Return. All normal facilities for editing, moving the cursor over an old command and re-entering it, etc are available. ▶

Some examples of use are:

/J.A.SMITH ;TEL 01 200 0222 (return)	..... enters the name and telephone number in the address book. The name must be followed by a space.
@J.A.SMITH ;TEL 01 200 0223 (return)	..... searches for the entry starting with J.A.SMITH and changes the telephone number. If there are two names with the same initials and surname, the first is acted on.
*J.A.SMITH ;TEL 01 200 0223 (return)	..... will delete the entry from the address book. The command must have exactly the same entry as that in the address book. Ie *J.A.SMITH (return) will have no effect here.
LOAD (return)	..... will find the first address book on tape, whatever it is called and load that into memory.
MERGE FAMILY (return)	..... searches for the address book FAMILY and merges the entries in it with the ones in memory. It is always faster to merge a small address book into a large one than *vice-versa*.
PRINT (return)	Outputs the address book to the printer.

An obvious application for the program is to store names and addresses so that you can print address labels quickly. To do this, you need to be able to print a (return) after every line of an entry and perhaps a (comma) in the entry. To let you do this, the program will interpret the character '/' when it occurs inside an entry as (comma)-(return) and ';' as (comma). So, to have the entry: J.A.SMITH, 1 ELM TREE AVENUE, BIRMINGHAM B1 A 2BD printed out in a normal address format, it should be entered as:

/J.A.SMITH /1; ELM TRE E AVENUE/BIRMINGHAM B1 A 2BD(return).

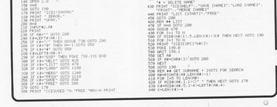
This will result in:

J.A.SMITH ,
1, ELM TREE AVENUE,
BIRMINGHAM B1A 2BD
being printed on the printer.

To provide more compatibility with VIC's normal operation, the Stop key is disabled by the command POKE 788, 194 in line 20. Then, when LISTing, the Stop key can be used as normal to stop the listing. The key is reactivated by POKE 788, 191 in line 110.

Error messages produced by the program are:

SYNTAX ERROR — the command you used was not recognised by the program.
BOOK FULL — there are no entries unused.
NAME NOT FOUND — you specified a name for alteration or deletion that was not in the address book.



```
100 PRINT "[CLS]**** ADDRESS BOOK ****"
110 MAX=100
120 POKE 788,194:REM ** KILL 'STOP' KEY
130 DIM N$(MAX)
140 OPEN 2,0
150 N=0
160 GOTO 190
170 PRINT "[CD]!SYNTAX"
180 PRINT " ERROR."
190 PRINT "OKAY."
200 A=0
210 INPUT#2,A$
220 PRINT
230 IF A$="" GOTO 200
240 K$=LEFT$(A$,1)
250 IF K$="/" THEN GOSUB 730:GOTO 200
260 IF K$="@" THEN A=-1:GOTO 850
270 IF K$="*" GOTO 850
280 K$=LEFT$(A$,4)
290 IF K$="END" THEN POKE 788,191:END
300 IF K$="HELP" GOTO 410
310 IF K$="LIST" GOTO 470
320 IF K$="LOAD" GOTO 1100
330 IF K$="MERG" GOTO 1340
340 IF K$="PRIN" GOTO 1210
350 IF K$="SAVE" GOTO 980
360 IF K$="FREE" GOTO 380
370 GOTO 170
380 PRINT "[CD]USED:"N,"FREE:"MAX-N:PRINT
    "[CD]BYTES FREE:"FRE(I)
390 GOTO 200
400 REM ** HELP
410 PRINT "[CD][REV]COMMANDS:[OFF][CD]"
420 PRINT "/ = ENTER NAME","@ = ALTER NAME",
    "* = DELETE NAME"
430 PRINT "[CD]HELP",,"SAVE [NAME]","LOAD [NAME]",
    "PRINT",,"MERGE [NAME]"
440 PRINT "LIST [START]","FREE"
450 GOTO 200
460 REM ** LIST
470 IF N=0 GOTO 200
480 GOSUB 670
490 FOR I=1 TO N
500 IF MID$(N$(I),5,LEN(K$))<K$ THEN NEXT:GOTO 190
510 FOR J=I TO N
520 PRINT "[CD][SPC]"N$(J)
530 POKE 198,0
540 WAIT 198,1
550 GET A$
560 IF A$=CHR$(3) GOTO 200
570 NEXT
580 GOTO 190
590 REM ** GET SURNAME + INITS FOR SEARCH
600 A$=RIGHT$(A$,LEN(A$)-1)
610 FOR I=5 TO LEN(A$)
620 IF MID$(A$,I,1)<>" " THEN NEXT:GOTO 170
630 K$=MID$(A$,5,I-4)+LEFT$(A$,4)
640 K=LEN(K$)-4
```

```
650 RETURN
660 REM ** GET [NAME] AFTER COMMAND EG. LOAD [NAME]
670 K$=""
680 FOR I=5 TO LEN(A$)
690 IF MID$(A$,I,1)<>" " THEN NEXT:RETURN
700 K$=RIGHT$(A$,LEN(A$)-I)
710 RETURN
720 REM ** ENTRY
730 GOSUB 600
740 N=N+1
750 IF N=1 THEN I=N:GOTO 820
760 IF N>MAX THEN N=MAX:PRINT "[CD]!BOOK FULL",,
    " ERROR.":RETURN
770 FOR I=1 TO N-1
780 IF MID$(N$(I),5,K)+LEFT$(N$(I),4)<K$
    THEN NEXT:GOTO 820
790 FOR J=N TO I STEP-1
800 N$(J)=N$(J-1)
810 NEXT
820 N$(I)=A$
830 RETURN
840 REM ** ALTER OR DELETE
850 GOSUB 600
860 IF N=0 GOTO 950
870 FOR I=1 TO N
880 IF MID$(N$(I),5,K)+LEFT$(N$(I),4)<>K$
    THEN NEXT:GOTO 950
890 IF A THEN N$(I)=A$:GOTO 200:REM **
    IF A<>0 THEN ALTER, ELSE DELETE
900 N=N-1
910 FOR J=I TO N
920 N$(J)=N$(J+1)
930 NEXT
940 GOTO 200
950 PRINT "[CD]!NAME NOT FOUND"
960 GOTO 180
970 REM ** SAVE
980 IF N=0 GOTO 200
990 GOSUB 670
1000 OPEN 1,1,1,K$
1010 PRINT#1,N
1020 PRINT "[CD]SAVING"N"ENTRIES[CD]"
1030 FOR I=1 TO N
1040 PRINT "[CU]"I
1050 PRINT#1,N$(I)
1060 NEXT
1070 CLOSE1
1080 GOTO 190
1090 REM ** LOAD
1100 GOSUB 670
1110 OPEN1,1,0,K$
1120 INPUT#1,N
1130 PRINT "[CD]LOADING"N"ENTRIES[CD]"
1140 FOR I=1 TO N
1150 PRINT "[CU]"I
1160 INPUT#1,N$(I)
1170 NEXT
1180 CLOSE1
1190 GOTO 190
1200 REM ** PRINT
1210 IF N=0 GOTO 200
1220 OPEN4,4
1230 FOR I=1 TO N
1240 PRINT#4
1250 FOR J=1 TO LEN(N$(I))
1260 IF MID$(N$(I),J,1)="/" THEN PRINT#4,",":GOTO 1290
1270 IF MID$(N$(I),J,1)=";" THEN PRINT#4,",";:GOTO 1290
1280 PRINT#4,MID$(N$(I),J,1);
1290 NEXT J,I
1300 PRINT#4
1310 CLOSE4
1320 GOTO 190
1330 REM ** MERGE
1340 GOSUB 730
1350 OPEN1,1,0,K$
1360 INPUT#1,N1
1370 PRINT "MERGING"N1"ENTRIES"
1380 IF N1+N>MAX THEN PRINT "[CD]!BOOK FULL":GOTO 180
1390 FOR I1=1 TO N1
1400 INPUT#1,A$
1410 A$="/"+A$
1420 GOSUB 730
1430 NEXT
1440 CLOSE1
1450 GOTO 190
```

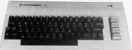
NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Eccleston Electronics announce 25% off all this VIC 20 software whilst stocks last

VIC	1212	VIC Programmers aid cartridge	£34.95
VIC	1213	VIC Machine code monitor cartridge	£34.95
VIC	1901	"Avenger"	£18.95
VIC	1902	"Star Battle"	£18.95
VIC	1904	"Super Slot"	£18.95
VIC	1905	"Jelly Monsters"	£18.95
VIC	1906	"Alien"	£18.95
VIC	1907	"Super Lander"	£18.95
VIC	1909	"Road Race"	£18.95
VIC	1910	"Rat Race"	£18.95
VIC	1912	"Mole Attack"	£18.95
VIC	1914	"Adventureland"	£24.95
VIC	1915	"Pirate Cove"	£24.95
VIC	1916	"Mission Impossible"	£24.95
VIC	1917	"The Count"	£24.95
VIC	1918	"Voodoo Castle"	£24.95
VIC	1919	"Sargon 2 Chess"	£24.95
VIC	1923	"Gorf"	£24.95
VIC	1924	"Omega Race"	£24.95
AUD	014	"Spiders of Mars"	£19.95
AUD	048	"Cloudburst"	£19.95
AUD	049	"Renaissance"	£19.95
AUD	050	"Satellites and Meteorites"	£19.95
AUD	051	"Meteor Run"	£19.95
AUD	067	"Trashman"	£19.95
AUD	068	"Tank Attack"	£19.95
AUD	069	"Outworld"	£19.95
AUD	072	"Astroblitz"	£19.95
CASSETTES			
VIC	2201	"Blitz"	£4.99
VIC	2202	"Hoppit"	£4.99
VIC	2203	"Race"	£4.99
VIC	2204**	"Strategic Advance"	£4.99
RAB		"Space Storm"	£6.99
RAB		"Orbis"	£9.99
RAB		"Annihilator"	£9.99
RAB		"Galactic Crossfire"	£9.99
RAB		"Quackers"	£9.99
RAB		"Alien Soccer"	£9.99
RAB		"Kreil"	£9.99
RAB	*	"Myriad"	£9.99
RAB	*	"Night Crawler"	£9.99
RAB		"Space Phreeks"	£9.99
RAB		"Hopper"	£9.99
RAB		"Adventure Pack 1"	£12.99
RAB		"Skramble"	£9.99
AUD	010	"Amok"	£6.95
AUD	026	"Alien Blitz"	£7.95
AUD	046	"Vic Trap/Sea Wolf/Bounce Out"	£8.95
AUD	053*	"Kosmic Kamikaze"	£7.95
AUD	055*	"Golf"	£7.95
AUD	063**	"Boss"	£14.95
AUD	064**	"Bonzo"	£7.95
AUD	071	"Pit"	£9.99
ROM	1001	"Martian Raider"	£9.99
ROM	1002	"Multisound Synthesizer"	£9.99
ROM	1003	"Shark Attack"	£9.99
ROM	1004	"Mind Twisters"	£9.99
ROM	1006	"Sea Invasion"	£9.99
ROM	1007*	"Moons of Jupiter"	£9.99
ROM	1008	"Space Attack"	£9.99
EDUCATION ('O' LEVEL)			
VIC	3401**	English Language	£9.99
VIC	3402**	Mathematics 1	£9.99
VIC	3403**	Mathematics 2	£9.99
VIC	3404**	Biology	£9.99
VIC	3405**	Chemistry	£9.99
VIC	3406**	Physics	£9.99
VIC	3407**	Computer Studies	£9.99
VIC	3408**	Geography	£9.99
VC	3409**	History	£9.99
VIC	3412**	General Knowledge (9-11 years old)	£9.99
VIC	3420*	Junior Maths — Apple Tree	£4.99
VIC	3421*	Junior Maths — Engine Shed	£4.99
VIC	3422*	Junior Maths — Lighthouse & Subtract	£4.99
HOME — Cassette Based			
VIC	3501**	"Quizmaster"	£9.99
VIC	3502**	"Know Your Own IQ"	£9.99
VIC	3503**	"Know Your Childs IQ"	£9.99
VIC	3504**	"Know Your Own Personality"	£9.99
VIC	3505**	"Robert Carrier's Menu"	£9.99
VIC	3506**	"VIC Money Manager"	£9.99
VIC	3507**	"VIC Road User"	£9.99
VIC	3510**	BBC "Ask the Family"	£9.99

KEY *............3K RAM required **..............8K RAM required

Commodore 64 and Cassette Deck in stock. Ring us for our competitive prices



Announcing Speeder Software for the Commodore 64:

Machine Code Monitor — Tape Programmers Toolkit —
Cat and Mouse Game

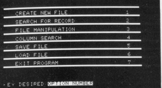
RING US TODAY ON 021-236-6220/1226
BARCLAYCARD & ACCESS ACCEPTED

ECCLESTON ELECTRONICS
Eccleston & Hart Limited,
8 Legge Lane, BIRMINGHAM B1 3LG.

A P Stephenson

MULTICOLUMN RECORDS

A fully updated version of the Multipurpose program with greatly increased facilities. Ideal for small business or personal information.



```
PRIMARY OPTIONS

CREATE NEW FILE                 1
SEARCH FOR RECORD               2
FILE MANIPULATION               3
COLUMN SEARCH                   4
SAVE FILE                       5
LOAD FILE                       6
EXIT PROGRAM                    7

KEY DESIRED OPTION NUMBER
```

This issue contains a program (of mine) called 'Multipurpose Records'. Each record was limited in length to the width of one screen line, allowing a practical maximum of four columns. It would have been easy to replace this arrangement with a system allowing one screen page per record but the advantage of more space would have meant the loss of column comparison. Running the eye down a column of data can provide valuable secondary information. In order to combine the advantages of both, a fresh approach was needed and this program is the result.

It allows a choice of serial or parallel (column or page) presentation of each record in a file. It was achieved by presenting the KEYFIELD as a stationary item but allowing each column to be revolved into view from the left or right. Once the desired column appears in the window, the file can be sequenced up or down through the various records. Alternatively, a complete record can be displayed in full page detail when required. Although there is no absolute restriction on the number of columns in each record, the file array has been dimensioned for a maximum of 10 columns in order to keep the memory cost down. While on the subject of memory it would be fair to mention that the program consumes an embarrassing amount of it. As it stands, it will not reside in an 8K PET. However, the screen messages lack the staccato shorthand often found in silicon vocal chords. The REM statements are sprinkled liberally and are equally verbose. It would be easy to get rid of the REMs, cut down on the textual material and slice out the disc-SAVE and disc-LOAD lines if unwanted. Extensive surgery of this kind could eventually slim the program down to 8K capacity although the residual memory would not hold many records.

Before keying in a program of this length, it would be wise to examine the facilities offered and judge whether or not the labour involved would be justified. This information could, of course, be gleaned by study of the listing but in the interests of personal pride the following commercial break may be of interest.

PRIMARY OPTIONS

Create File enables a new file to be set up and the column headings and date entered. After each record is complete, the amount of memory left is displayed ...a necessary warning to deter those of a garrulous nature.

Save File can be used to store on cassette tape or disc with either drive '0' or drive '1' choice. **Load File** is the complementary function.

Search for Record allows any individual record to be accessed by asking for the key field or the record number.

Column Search can be used to examine the entire file and output the key field and record number of all records which have parameters equal to or within a given range of the search parameter. For example, if the file was on transistor specifications, it is possible to ask for, say, all transistors with a power-max rating less than 600 mW. Similarly, a file on employees may be examined for those under the age of 50 etc.

File Manipulation is a subset of the Primary Options and is examined later.

Exit Program although superficially a trivial option is necessary because of the RUN/STOP key is inhibited at the start of the program with POKE 144,49 (POKE 144,88 for BASIC 4 users). The program is therefore locked in an endless loop until the Exit Program option is executed and the RUN/STOP is released with POKE 144,46 (POKE 144,85 for BASIC 4 users).

FILE MANIPULATIONS

Twelve options are available (if we include the return to **Primary Options** as one of them).

View Columns allows the various columns to be revolved into the window. Key '<' will revolve left and key '>' right.

Next Record slides the next record into view (at least the key-filled and the chosen column). **Last Record**, as before but moving backwards.

Modify Data allows the data belonging to the column shown to be changed.

Modify Column Heading will not be a frequent requirement but was included (after its original omission) because of criticism from a colleague; apparently he makes frequent mistakes when creating a new file!

Modify Key Field allows changes to the key field. This is the one that doesn't move when the columns are rotated.

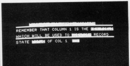
Change File Name is handy if a file has been loaded from disc and modified in some way.

Add Record allows additional records to be included in the file and the next highest record number allocated.

View Record displays the complete record with all its columns.

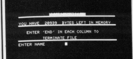
View File is a simple scrolling action of all records. Pressing the 'Space Bar' at any time during the scroll will return control to the window display mode with the particular record captured.

Totalise Column causes the



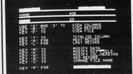
```
REMEMBER THAT COLUMN 1 IS THE [illegible]
WHICH WILL BE USED TO [illegible] RECORD
STATE [illegible] OF COL 1
```

Taking the first steps in creating a new file.



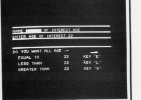
```
YOU HAVE  28939  BYTES LEFT IN MEMORY

   ENTER 'END' IN EACH COLUMN TO
         TERMINATE FILE
ENTER NAME
```

Once all the required column headings have been set up information can be entered as records.



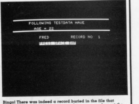
```
NAME                      AGE
                          22
USE '<' AND '>' TO        VIEW COLUMNS
KEY 'R' TO                VIEW RECORDS
KEY 'F' TO                VIEW FILE

KEY 'T' TO                TOTALISE COLUMN
KEY 'N' FOR               NEXT RECORD
KEY 'L' FOR               LAST RECORD

KEY 'M' TO                MODIFY DATA
KEY 'K' TO                MODIFY KEY FIELD
KEY 'H' TO                MODIFY COL.HEADING
KEY 'A' TO                ADD RECORD
KEY 'C' TO                CHANGE FILE NAME

KEY 'P' FOR
```

Once the file is completed a secondary option menu is provided to allow you to manipulate the information.



```
NAME [illegible] OF INTEREST AGE
ENTER AGE OF INTEREST 22

DO YOU WANT ALL AGE :-
  EQUAL TO            22      KEY 'E'
  LESS THAN           22      KEY 'L'
  GREATER THAN        22      KEY 'G'
```

We've decided to search the file for any records where the recorded age is 22. The program also allows us to perform a variety of selective searches at this stage.



```
      FOLLOWING TESTDATA HAVE
       AGE = 22
        FRED          RECORD NO  1
        PRESS SPACE BAR
```

Bingo! There was indeed a record buried in the file that matched our search parameters and we can now inspect the rest of the information stored for Fred.

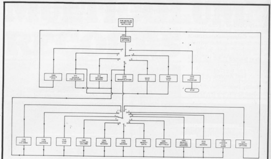

DIM A$(50,10) INHIBIT STOP INITIALISE
220
PRIMARY OPTIONS
CREATE FILE
SEARCH FOR RECORD
COLUMN SEARCH
FILE MANIPULATION
SAVE FILE
LOAD FILE
EXIT PROGRAM
STOP
VIEW COLUMNS
VIEW RECORDS
VIEW FILE
TOTALISE COLUMN
NEXT RECORD
LAST RECORD
MODIFY DATA
MODIFY KEY FIELD
MODIFY COLUMN HEADING
ADD RECORD
CHANGE FILE NAME
PRIMARY OPTIONS

Fig. 1. The structure diagram of the program showing the links and program line numbers of the various options.

column in the window to be totalised and the average displayed. Naturally the facility is of use only if the data is purely numeric.

MODIFICATIONS

The program has been tried out by certain colleagues of mine (not normally noted for kindness and tact) who have reluctantly declaired it to be 'not bad'... which is indeed high praise from them. The listing shown is aided by a structure chart ('structure' in this sense relating to the normal English without Dijkstra overtones). The switches shown are, of course, intended to represent the software kind, the top one being the ON...GOTO statement in line 450 and the bottom one the set of IF...THEN statements in lines 1170 to 1310 inclusive. The DIMENSION statement appears three times, lines 180, 4620 and 5320. This was necessary because when loading a file from tape or disc of smaller dimensions than the previously existing file, the residue would have remained. This is prevented by the CLR statement which precedes the DIM. The listing shows the actual dimension statement to be DIM A$(50,10) allowing 50 records, each of 10 columns. This is a purely arbitrary choice and depending on the available memory, can be increased to the limit... but remember to change all three.

PROGRAM PORTABLITY

The program was written for the PET series with the New ROM (revision 3). There are only two POKEs which may have to be changed if run on Old ROMs...POKE 158,0 which occurs many times should be changed to POKE 525,0. The other danger is the inhibit STOP...POKE 144,49 which is best left out altogether in old ROMs. With regard to using the program on machines other than the PET series, apart from the POKEs, the BASIC is fairly standard and should require only trivial adjustments.

```
100 X=0:REM ** NUMBER OF COLUMNS
110 Y=0:REM ** NUMBER OF RECORDS
120 R=0:REM ** ROW
130 C=0:REM ** COLUMN
140 I$="":REM ** KEYBOARD INPUT
150 K$="":REM ** GET INPUT
160 F8$="":REM ** MESSAGE FLASH
170 N$="":REM ** FILE NAME
180 DIM A$(50,10):A$(R,C)="":REM ** FILE
190 POKE 144,49:REM ** INHIBIT STOP KEY, POKE 144,88 FOR BASIC 4
200 A$(R,C)="":M=15359:F=0:S=0
210 REM ** OPTIONS
220 PRINT CHR$(147)
230 PRINT TAB(12)"[REV]PRIMARY OPTIONS[OFF]"
240 PRINT:PRINT:PRINT
250 GOSUB 3830
260 PRINT TAB(3)"CREATE NEW FILE"TAB(35)"1"
270 GOSUB 3830
280 PRINT TAB(3)"SEARCH FOR RECORD"TAB(35)"2"
```

```
290 GOSUB 3830
300 PRINT TAB(3)"FILE MANIPULATION"TAB(35)"3"
310 GOSUB 3830
320 PRINT TAB(3)"COLUMN SEARCH"TAB(35)"4"
330 GOSUB 3830
340 PRINT TAB(3)"SAVE FILE"TAB(35)"5"
350 GOSUB 3830
360 PRINT TAB(3)"LOAD FILE"TAB(35)"6"
370 GOSUB 3830
380 PRINT TAB(3)"EXIT PROGRAM"TAB(35)"7"
390 GOSUB 3830
400 PRINT "KEY DESIRED [REV]OPTION NUMBER[OFF]"
410 POKE 158,0
420 GET K$
430 IF K$="" THEN 420
440 IF VAL(K$)<1 OR VAL(K$)>7 THEN 420
450 ON VAL(K$) GOTO 2080,470,910,2750,3860,4450,5530
460 REM ** SEARCH FOR RECORD
470 PRINT CHR$(147)
480 GOSUB 3830
490 IF Y=0 THEN F8$="[REV]FILE NOT RESIDENT ![OFF]"
500 GOSUB 3730
510 GOTO 220
520 PRINT TAB(11)"[REV]SEARCH PROCEDURES[OFF]"
530 PRINT:PRINT:PRINT
540 GOSUB 3830
550 PRINT:PRINT
560 PRINT "KEY 'K' TO SEARCH BY [REV]KEY FIELD[OFF]"
570 PRINT
580 PRINT "KEY 'N' TO SEARCH BY [REV]RECORD NUMBER[OFF]"
590 PRINT:PRINT
600 GOSUB 3830
610 POKE 158,0
620 GET K$
630 IF K$="" THEN 620
640 IF K$="K" THEN 670
650 IF K$="N" THEN 710
660 GOTO 620
670 PRINT TAB(3)"ENTER KEY FIELD"
680 PRINT:PRINT TAB(3);
690 GOSUB 3670
700 GOTO 790
710 PRINT "[2 SPC]THERE ARE "Y" RECORDS IN THIS FILE"
720 PRINT
730 PRINT TAB(3)"ENTER RECORD NUMBER"
740 PRINT:PRINT TAB(3);
750 GOSUB 3670
760 R=VAL(I8$)
770 IF R=0 OR R>Y THEN PRINT"[CU]":GOTO 730
780 GOTO 910
790 FOR R=1 TO Y
800 IF I8$=A$(R,1) THEN 910
810 NEXT R
820 PRINT CHR$(147)
830 PRINT TAB(240)
840 GOSUB 3830
850 PRINT TAB(10);"[REV]NO SUCH RECORD EXISTS ![OFF]"
860 GOSUB 3830
870 FOR Z=1 TO 1000
880 NEXT Z
890 GOTO 470
900 REM ** MANIPULATE FILE
910 PRINT CHR$(147):C=2
920 IF Y=0 THEN F8$="[REV]NO FILE EXISTS[OFF]":
    GOSUB 3730:GOTO 220
930 PRINT "[REV]";N$ TAB(24)"[REV]RECORD NO";R
940 GOSUB 3830
950 PRINT A$(0,1)TAB(20)A$(0,C+CN)
960 GOSUB 3830
970 PRINT "[REV]"A$(R,1);TAB(20)A$(R,C+CN)
980 GOSUB 3830
990 POKE 158,0
1000 PRINT "USE '<' AND '>' TO[3 SPC]VIEW COLUMNS"
1010 PRINT "KEY 'R' TO[11 SPC]VIEW RECORDS"
1020 PRINT "KEY 'F' TO[11 SPC]VIEW FILE"
1030 PRINT
1040 PRINT "KEY 'T' TO[11 SPC]TOTALISE COLUMN"
1050 PRINT "KEY 'N' FOR[10 SPC]NEXT RECORD"
1060 PRINT "KEY 'L' FOR[10 SPC]LAST RECORD"
1070 PRINT
1080 PRINT "KEY 'M' TO[11 SPC]MODIFY DATA"
1090 PRINT "KEY 'K' TO[11 SPC]MODIFY KEY FIELD"
1100 PRINT "KEY 'H' TO[11 SPC]MODIFY COL.HEADING"
1110 PRINT "KEY 'A' TO[11 SPC]ADD RECORD"
1120 PRINT "KEY 'C' TO[11 SPC]CHANGE FILE NAME"
1130 PRINT
1140 PRINT "KEY 'P' FOR[10 SPC][REV]PRIMARY OPTIONS[OFF]"
1150 POKE 158,0
1160 GET K$
1170 IF K$="" THEN 1160
1180 IF K$=">" THEN CN=CN+1
1190 IF K$="<" THEN CN=CN-1
1200 IF K$="N" THEN R=R+1:IF R>Y THEN R=Y:GOTO 910
1210 IF K$="L" THEN R=R-1:IF R<1 THEN R=1:GOTO 910
1220 IF K$="M" THEN 1340
1230 IF K$="P" THEN 220
1240 IF K$="F" THEN 1510
1250 IF K$="R" THEN 1790
1260 IF K$="T" THEN 1940
1270 IF K$="C" THEN 1450
1280 IF K$="H" THEN 1400
1290 IF K$="K" THEN CN=0:C=1:GOTO 1340
1300 IF K$="A" THEN F=1:GOTO 2510
1310 IF C+CN<2 OR C+CN>X THEN C=2:CN=0
1320 GOTO 910
1330 REM ** MODIFY RECORD
1340 GOSUB 3830
1350 PRINT "ENTER CORRECT DATA FOR [REV]"A$(0,C+CN)
1360 GOSUB 3670
1370 A$(R,C+CN)=I8$
1380 C=2
1390 GOTO 910
1400 GOSUB 3830
1410 PRINT "ENTER CORRECT COLUMN HEADING ";
1420 GOSUB 3670
1430 A$(0,C+CN)=I8$
1440 GOTO 910
1450 GOSUB 3830
1460 PRINT "ENTER NEW FILE NAME "
1470 GOSUB 3670
1480 N$=I8$
1490 GOTO 910
1500 REM ** VIEW FILE
1510 PRINT CHR$(147)
1520 GOSUB 3830
1530 PRINT TAB(9)"FILE WILL SCROLL DOWN"
1540 PRINT:PRINT
1550 PRINT TAB(4)"TO STOP SCROLLING,PRESS [REV]SPACE BAR[OFF]"
1560 GOSUB 3830
1570 FOR Z=1 TO 2000
1580 NEXT Z
1590 PRINT CHR$(147)
1600 PRINT TAB(12)"[REV]";N$"[OFF]"
1610 PRINT
1620 GOSUB 3830
1630 FOR R=1 TO Y
1640 FOR C=1 TO X
1650 PRINT A$(0,C)TAB(20)A$(R,C)
1660 IF C=1 THEN PRINT "[401D]"
1670 FOR Z=1 TO 400
1680 NEXT Z
1690 NEXT C
1700 GOSUB 3830
1710 GET K$
1720 IF K$="[SPC]" THEN 910
1730 NEXT R
1740 FOR Z=1 TO 600
1750 NEXT Z
1760 R=Y
1770 GOTO 910
1780 REM ** VIEW RECORD
1790 PRINT CHR$(147)
1800 PRINT "[REV]";N$ TAB(19)"RECORD NO ";R
1810 GOSUB 3830
1820 PRINT TAB(15)A$(R,1)
1830 GOSUB 3830
1840 FOR C=2 TO X
1850 PRINT A$(0,C)TAB(19)A$(R,C)
1860 GOSUB 3830
1870 NEXT C
1880 PRINT TAB(5)"[REV]PRESS SPACE BAR TO RETURN[OFF]"
1890 POKE 158,0
1900 GET K$
1910 IF K$<>"[SPC]" THEN 1900
1920 GOTO 910
1930 REM ** TOTALS
1940 PRINT CHR$(147)
1950 GOSUB 3830
1960 T=0
1970 PRINT
1980 FOR R=1 TO Y
1990 T=T+VAL(A$(R,C+CN))
2000 NEXT R
2010 PRINT "[REV]"A$(0,C+CN)"[OFF]COLUMN TOTAL IS"TAB(30);T
2020 PRINT
2030 PRINT "AND THE AVERAGE IS"TAB(30)T/Y
2040 GOSUB 3830
2050 PRINT:PRINT
2060 GOTO 1880
2070 REM ** CREATE NEW FILE
2080 PRINT CHR$(147)
2090 F8$="WARNING,ARE YOU SURE ?"
2100 GOSUB 3730
2110 PRINT TAB(129)"ANSWER Y(YES) OR N(NO)"
2120 POKE 158,0
2130 GET K$
2140 IF K$="" THEN 2130
2150 IF K$="N" THEN 220
2160 IF K$="Y" THEN 2180
2170 GOTO 2130
2180 PRINT CHR$(147)
2190 CLR
2200 DIM A$(50,10)
2210 PRINT TAB(12)"[REV]CREATE RECORD[OFF]"
2220 PRINT
2230 GOSUB 3830
2240 PRINT "WHAT IS FILE NAME [2↑K][SPC]";
2250 GOSUB 3670
2260 N$=I8$
2270 PRINT
2280 GOSUB 3830
2290 PRINT "HOW MANY COLUMNS IN EACH RECORD [2↑K][SPC]";
2300 GOSUB 3670
2310 X=VAL(I8$)
2320 PRINT
2330 IF X<1 OR X>10 THEN PRINT "[REV]MAXIMUM IS 10[OFF]":GOTO 2290
2340 PRINT CHR$(147)
2350 C=1
2360 PRINT "[REV]THIS FILE IS NAMED ";N$
2370 GOSUB 3830
2380 PRINT "REMEMBER THAT COLUMN 1 IS THE [REV]KEY FIELD[OFF]"
2390 PRINT
2400 PRINT "WHICH WILL BE USED TO [REV]IDENTIFY[OFF] RECORDS"
2410 GOSUB 3830
2420 PRINT "STATE [REV]TITLE[OFF] OF COL"C" [2↑K]";
```

```
2430 GOSUB 3670
2440 A$(0,C)=I8$
2450 PRINT
2460 GOSUB 3830
2470 IF C=X THEN 2510
2480 C=C+1
2490 GOTO 2420
2500 REM ## ENTER FILE DATA
2510 IF F=1 THEN R=Y+1:Y=Y+1:GOTO 2530
2520 R=1:Y=1
2530 PRINT CHR$(147)
2540 PRINT TAB(12)"[REV]RECORD NUMBER";R"[OFF]"
2550 GOSUB 3830
2560 PRINT:PRINT "YOU HAVE "FRE(0)" BYTES LEFT IN MEMORY"
2570 GOSUB 3830
2580 PRINT
2590 PRINT TAB(3)"ENTER 'END' IN EACH COLUMN TO"
2600 PRINT
2610 PRINT TAB(9)"TERMINATE FILE"
2620 GOSUB 3830
2630 FOR C=1 TO X
2640 PRINT "ENTER "A$(0,C)TAB(18);
2650 GOSUB 3670
2660 A$(R,C)=I8$
2670 PRINT
2680 GOSUB 3830
2690 NEXT C
2700 IF F=1 THEN F=0:GOTO 910
2710 IF I8$="END" THEN R=1:Y=Y-1:GOTO 220
2720 R=R+1:Y=Y+1
2730 GOTO 2530
2740 REM ## COLUMN SEARCH
2750 PRINT CHR$(147)
2760 PRINT TAB(240)
2770 GOSUB 3830
2780 E=0
2790 FOR R=1 TO Y
2800 PRINT "NAME [REV]COLUMN[OFF] OF INTEREST ";
2810 GOSUB 3670
2820 CI$=I8$
2830 PRINT
2840 GOSUB 3830
2850 C=1
2860 IF A$(0,C)=CI$ THEN 2930
2870 C=C+1:IF C<X+1 THEN 2860
2880 IF C<X+1 THEN 2860
2890 PRINT CHR$(147)
2900 F8$="NO SUCH COLUMN"
2910 GOSUB 3730
2920 GOTO 220
2930 PRINT "ENTER ";CI$;" OF INTEREST ";
2940 GOSUB 3670
2950 DI$=I8$
2960 PRINT
2970 GOSUB 3830
2980 PRINT:PRINT
2990 GOSUB 3830
3000 IF VAL(LEFT$(DI$,1))=0 THEN 3430
3010 PRINT"DO YOU WANT ALL ";CI$;" :-"
3020 PRINT
3030 PRINT TAB(2)"EQUAL TO"TAB(20)DI$ TAB(28)"KEY 'E'"
3040 PRINT
3050 PRINT TAB(2)"LESS THAN"TAB(20)DI$ TAB(28)"KEY 'L'"
3060 PRINT
3070 PRINT TAB(2)"GREATER THAN"TAB(20)DI$ TAB(28)"KEY 'G'"
3080 POKE 158,0
3090 GET K$
3100 IF K$="" THEN 3090
3110 IF K$="E" THEN 3430
3120 IF K$="L" THEN 3150
3130 IF K$="G" THEN 3290
3140 GOTO 3090
3150 PRINT CHR$(147)
3160 GOSUB 3830
3170 S=0
3180 PRINT TAB(6)"FOLLOWING "N$" HAVE "
3190 PRINT
3200 PRINT TAB(8)CI$" LESS THAN "DI$
3210 GOSUB 3830
3220 PRINT
3230 FOR R=1 TO Y
3240 IF VAL(A$(R,C))<VAL(DI$) THEN PRINT TAB(10)A$(R,1)TAB(22)"RECORD NO ";R
3250 S=1
3260 NEXT R
3270 IF S=0 THEN PRINT CHR$(147):GOTO 3560
3280 GOTO 3590
3290 PRINT CHR$(147)
3300 GOSUB 3830
3310 S=0
3320 PRINT TAB(6)"FOLLOWING "N$" HAVE "
3330 PRINT
3340 PRINT TAB(8)CI$" GREATER THAN "DI$
3350 GOSUB 3830
3360 PRINT
3370 FOR R=1 TO Y
3380 IF VAL(A$(R,C))>VAL(DI$) THEN PRINT TAB(10)A$(R,1)TAB(22)"RECORD NO ";R
3390 S=1
3400 NEXT Y
3410 IF S=0 THEN PRINT CHR$(147):GOTO 3560
3420 GOTO 3590
3430 PRINT CHR$(147)
3440 GOSUB 3830
3450 S=0
3460 PRINT TAB(6)"FOLLOWING "N$" HAVE "
3470 PRINT
3480 PRINT TAB(8)CI$" = "DI$
3490 GOSUB 3830
```

```
3500 PRINT
3510 FOR R=1 TO Y
3520 IF A$(R,C)=DI$ THEN PRINT TA
     "RECORD NO ";R
3530 S=1
3540 NEXT R
3550 IF S=1 THEN 3590
3560 F8$="[REV]NO DATA EXISTING
3570 GOSUB 3730
3580 GOTO 910
3590 PRINT
3600 POKE 158,0
3610 PRINT TAB(10)"[REV]PRESS SP
3620 GET K$
3630 IF K$="" THEN 3620
3640 IF K$="[SPC]" THEN R=R-1:GOT
3650 GOTO 3620
3660 REM ## CRASH-PROOF INPUT TO
3670 OPEN 1,0
3680 INPUT#1,I8$
3690 IF I8$="" THEN 3680
3700 CLOSE 1
3710 RETURN
3720 REM ## FLASH F8$
3730 FOR Z=1 TO 6
3740 PRINT TAB(10)F8$
3750 FOR T8=1 TO 250
3760 NEXT T8
3770 PRINT CHR$(147)
3780 FOR T8=1 TO 100
3790 NEXT T8
3800 NEXT Z
3810 RETURN
3820 REM ## PRINT LINE
3830 PRINT "[40↑#]"
3840 RETURN
3850 REM ## SAVE FILE
3860 PRINT CHR$(147)
3870 IF Y=0 THEN F8$="[REV]NO FI
     GOSUB 3730:GOTO 220
3880 PRINT TAB(240)
3890 GOSUB 3830
3900 PRINT
3910 PRINT TAB(3)"ARE YOU SAVING
     [REV]TAPE[OFF]?"
3920 PRINT
3930 GOSUB 3830
3940 PRINT:PRINT:PRINT
3950 PRINT TAB(11)"KEY 'D' OR 'T
3960 POKE 158,0
3970 GET K$
3980 IF K$="" THEN 3970
3990 IF K$="T" THEN 4030
4000 IF K$="D" THEN 5000
4010 GOTO 3970
4020 REM ## SAVE FILE ON TAPE
4030 PRINT CHR$(147)
4040 PRINT TAB(240)
4050 PRINT TAB(120)
4060 PRINT "HAVE YOU INSERTED A
4070 PRINT
4080 PRINT "HAVE YOU REWOUND IT
4090 PRINT
4100 PRINT "HAVE YOU SWITCHED MO
4110 PRINT:PRINT
4120 PRINT "[REV]YOU MUST NOW AN
4130 POKE 158,0
4140 GET K$
4150 IF K$<>"Y" THEN 4140
4160 PRINT CHR$(147)
4170 PRINT TAB(240)
4180 GOSUB 3830
4190 PRINT TAB(12)"[REV]BE PATIE
4200 GOSUB 3830
4210 PRINT:PRINT
4220 PRINT TAB(5)"YOUR FILE [REV
4230 PRINT
4240 PRINT TAB(5)"WILL TAKE TIME
4250 PRINT:PRINT
4260 OPEN 1,1,1,N$
4270 PRINT#1,Y
4280 PRINT#1,X
4290 FOR R=1 TO Y
4300 FOR C=1 TO X
4310 PRINT#1,A$(R,C)
4320 NEXT C
4330 NEXT R
4340 FOR C=1 TO X
4350 PRINT#1,A$(0,C)
4360 NEXT C
4370 CLOSE 1
4380 GOSUB 3830
4390 PRINT "FILE NAMED [REV]";N
4400 GOSUB 3830
4410 FOR Z=1 TO 1000
4420 NEXT Z
4430 GOTO 220
4440 REM ## LOAD FILE
4450 PRINT CHR$(147)
4460 PRINT TAB(240)
4470 GOSUB 3830
4480 PRINT TAB(2)"ARE YOU LOADIN
4490 GOSUB 3830
4500 PRINT:PRINT
4510 PRINT TAB(5)"KEY 'T' OR 'D'"
4520 POKE 158,0
INT TAB)A$(R,1)TAB(22)
STING[OF"
ESS SPACBAR[OFF]"
=-1:GOTO10
UT TO 1
NO FILEXISTS[OFF] ":
SAVING O[REV]DISC[OFF] OR
 OR 'T''
APE
TED A CRETTE?"
ND IT TOESIRED POSITION"
HED MOTOOFF?"
NOW ANSW Y(YES)[OFF]"
 PATIENT[OFF]"
E [REV]"#:"[OFF]"
E TIME TSAVE ON TAPE"
V]";N$"[FF]NOW ON TAPE"
LOADING ROM TAPE OR DISC ?"
OR 'D'"
```

```
4530 GET K$
4540 IF K$="" THEN 4530
4550 IF K$="T" THEN 4590
4560 IF K$="D" THEN 5290
4570 GOTO 4530
4580 REM ** LOAD FROM TAPE
4590 PRINT CHR$(147)
4600 PRINT TAB(240)
4610 CLR
4620 DIM A$(50,10)
4630 PRINT "ENTER FILE NAME";TAB(20);
4640 GOSUB 3670
4650 N$=I$
4660 PRINT
4670 PRINT TAB(240)
4680 PRINT "IS THIS FILE IN CASSETTE?"
4690 PRINT
4700 PRINT "IS IT REWOUND?"
4710 PRINT
4720 PRINT "IS THE MOTOR SWITCHED OFF?"
4730 PRINT:PRINT:PRINT
4740 PRINT "IF SO,ANSWER Y(YES)"
4750 POKE 158,0
4760 GET K$
4770 IF K$="" THEN 4760
4780 IF K$<>"Y" THEN 4760
4790 PRINT CHR$(147)
4800 PRINT TAB(240)
4810 GOSUB 3830
4820 PRINT TAB(3)"HAVE PATIENCE ! THIS TAKES TIME"
4830 GOSUB 3830
4840 PRINT
4850 OPEN1,1,0,N$
4860 INPUT#1,Y
4870 INPUT#1,X
4880 FOR R=1 TO Y
4890 FOR C=1 TO X
4900 INPUT#1,A$(R,C)
4910 NEXT C
4920 NEXT R
4930 FOR C=1 TO X
4940 INPUT#1,A$(0,C)
4950 NEXT C
4960 CLOSE1
4970 POKE 144,49:REM ** SEE LINE 190
4980 GOTO 470
4990 REM ** SAVE FILE ON DISC
5000 PRINT CHR$(147)
5010 PRINT TAB(240)
5020 GOSUB 3830
5030 PRINT TAB(2)"DO YOU WANT TO SAVE ON [REV]DRIVE '0'[OFF]"
5040 PRINT
5050 PRINT TAB(2)"OR ON [REV]DRIVE '1'[OFF]?"
5060 GOSUB 3830
5070 PRINT:PRINT:PRINT
5080 GET K$
5090 IF K$="" THEN 5080
5100 IF K$="0" THEN DR=0:GOTO 5130
5110 IF K$="1" THEN DR=1:GOTO 5130
5120 GOTO 5080
5130 N1$="DR:":N2$=",SEQ,WRITE"
5140 NN$=N1$+N$+N2$
5150 OPEN1,8,4,NN$
5160 PRINT#1,Y;CHR$(13);
5170 PRINT#1,X;CHR$(13);
5180 FOR R=1 TO Y
5190 FOR C=1 TO X
5200 PRINT#1,A$(R,C);CHR$(13);
5210 NEXT C
5220 NEXT R
5230 FOR C=1 TO X
5240 PRINT#1,A$(0,C);CHR$(13);
5250 NEXT C
5260 CLOSE1
5270 GOTO 220
5280 REM ** LOAD FILE FROM DISC
5290 PRINT CHR$(147)
5300 PRINT TAB(240)
5310 CLR
5320 DIM A$(50,10)
5330 PRINT "ENTER [REV]NAME[OFF] OF FILE"
5340 PRINT:PRINT:PRINT:PRINT TAB(7)
5350 GOSUB 3670
5360 N$=I$:NN$=N$
5370 N1$="0:":N2$=",SEQ,READ"
5380 NN$=N1$+NN$+N2$
5390 OPEN1,8,4,NN$
5400 INPUT#1,Y
5410 INPUT#1,X
5420 FOR R=1 TO Y
5430 FOR C=1 TO X
5440 INPUT#1,A$(R,C)
5450 NEXT C
5460 NEXT R
5470 FOR C=1 TO X
5480 INPUT#1,A$(0,C)
5490 NEXT C
5500 CLOSE1
5510 GOTO 470
5520 REM ** EXIT
5530 PRINT CHR$(147)
5540 F8$="[REV]P R O G R A M   E X I T[OFF]"
5550 GOSUB 3730
5560 POKE 144,46:REM ** POKE 144,85 FOR BASIC 4
5570 END
```

Listing 1. Program to produce multicolumn records.

An adjustment may be required in line 200 depending on memory size. For 16K PETs, no change is required. For an 8K system, M=15359 should be changed to M=7167. For a 32K PET, it should be M=31743. This value is used when creating a file to warn how much memory is left after each entry.

The INPUT subroutine at line 3360 to 3710 is peculiar to the Commodore PET and can be replaced by simple INPUT for most other types.

KEYING IT IN

It is always a daunting task to key in a long program such as this. Some people just sit down and keep bashing the keys relentlessly until it is finished . . . a Herculean task. I would never have the courage for this because there would almost certainly be multiple mistakes. My plan (being a pessimistic, cautious type of person) would be based on the modular system. Enter about 10 or 20 lines at a time, stopping at some logical module-end and stick in a temporary STOP then RUN the program to that point to see the results. For example, enter lines 100 to 450 inclusive which covers the 'Primary Options' page and put STOP at lines 2080, 470, 910, 2750, 3860, 4450 and 5530. When this is RUN and the various options tried out, it is easy to check that the correct linkage from the ON . . . GOTO statement is established. Then, proceed to enter each primary option part separately and RUN before proceeding with the next one. One final warning — keep on loading each module on to tape (or disc) as each part is proved. In this way the tape will gradually grow and will act as an insurance policy if you do something daft during the current entering session.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

THE GAMES YOU'VE BEEN DYING TO SEE!

For the Unexpanded VIC-20	
TVIC 3 Skramble!	£7.95
TVIC 4 Terminal Invaders	£5.95
TIVC 5 Meteor Blaster	£5.95
This Month Only, Reduced from £7.95	
TVIC 6 Gridder	for just £4.95
TVIC 7 Line Up 4/Reversi	£7.95
TVIC 8 Get Lost! (3D Maze)	£5.95
Adventures for VIC-20 with 16K Expansion	
TVIC 9 The Curse of the Werewolf	£9.95
TVIC 10 Rescue from Castle Dread	£9.95
Commodore 64 Software	
TC641 Super Skramble	£9.95
For the Dragon 32	
TDRAG 1 Line Up 4	£4.95

DEALER ENQUIRIES WELCOME

Machine-code programmers wanted! We will pay up to £1000 for good, original programs for any of the popular micros.

Demand our games at all good computer shops or buy mail order from:

TERMINAL SOFTWARE, DEPT. PST
28 Church Lane, Prestwich, Manchester M25 5AJ

MAIL ORDER PROTECTION SCHEME

If you order goods from Mail Order Advertisers in this magazine and pay by post in advance of delivery, this publication will consider you for compensation if the advertiser should become insolvent or bankrupt, provided:

1. You have not received the goods or had your money returned; and
2. You write to the publisher of this publication explaining the position not earlier than 28 days from the day you sent your order and not later than 2 months from that day.

Please do not wait until the last moment to inform us. When you write, we will tell you how to make your claim and what evidence of payment is required.

We guarantee to meet claims from readers made in accordance with the above procedure as soon as possible after the advertiser has been declared bankrupt or insolvent to a limit of £1,800 per annum for any one advertiser, so affected, and up to £5,400 p.a. in respect of all insolvent advertisers. Claims may be paid for higher amounts, or when the above procedures have not been complied with, at the discretion of this publication, but we do not guarantee to do so in view of the need to set some limit to this commitment and to learn quickly of reader's difficulties.

This guarantee covers only advance payment sent in direct response to an advertisement in this magazine (not, for example, payments made in response to catalogues, etc, received as a result of answering such advertisements):

CLASSIFIED ADVERTISEMENTS ARE EXCLUDED.

GIVE YOUR VIC20 & 64 IEEE PLUS RS232

VIC and 64 users

Would you like to be able to access **any** of these peripherals from your computer?

- ⅓ megabyte disks (Commodore 4040 drive)
- 1 megabyte disks (Commodore 8050 drive)
- 10 megabyte disks (Commodore 9090 hard disk)
- Printers including a wide range of inexpensive IEEE and RS232 matrix and quality printers
- IEEE instruments such as volt meters, plotters etc.

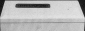
Now you are no longer limited by the VIC or the 64's serial bus. Simply by attaching INTERPOD you can vastly increase the power of your VIC 20 and when used with the new 64, INTERPOD turns the computer into a really powerful system.

With INTERPOD the VIC and 64 become capable of running really professional quality software such as Word-processing, Accounting, Instrument control and many more.

INTERPOD will work with any software. No extra commands are required and INTERPOD does not affect your computer in any way.

Using INTERPOD is as easy as this:

Simply plug INTERPOD into the serial port of your computer, power-up and you are ready to communicate with any number of parallel and serial IEEE devices and any RS232 printer.



INTERPOD

£125

Oxford Computer Systems (Software) Ltd.
Hensington Road, Woodstock, Oxford OX7 1JR, England Tel. (0993) 812700

UTILITIES

Subroutine Library — A library of BASIC subroutines.
Originally published in **Computing Today**, October 1982.

Toolkit Program — A simple toolkit program for the Commodore 64.

Tailoring VIC's Characters — Create your own characters on your VIC-20.
Originally published in **Computing Today**, February 1983.

Maxi-Mander — Bomb-proofing your software against unskilled fingers.
Originally published in **Computing Today**, July 1981.

VIC Blow Up — Find out how characters are made up and generate giant version on the VIC-20.
Originally published in **Computing Today**, January 1983.

Program Protection — Simple tips on how to protect your programs from being easily copied.
Originally published in **Computing Today**, June 1982.

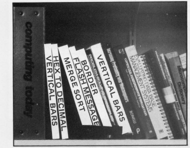
Bibliography — A brief perusal of some of the multitude of books on Commodore computers.

PET PERSONAL COMPUTER GUIDE
Herbert D. Peckham
Hands-On Basic with a PET
PET
Mastering the VIC-20
A.J.Jones E.A.Coley D.G.J.Cole
Garry Marshall
LEARNING TO USE THE PET COMPUTER
VIC GRAPHICS
NICK HAMPSHIRE
PET GAMES AND RECREATIONS
Mac Oglesby
Len Lindsay
Dorothy Kunkin

A P Stephenson

SUBROUTINE LIBRARY

A 'starter kit' of general purpose routines to make programming easier.



As far as BASIC is concerned, a subroutine is something you 'call up' with the keyword GOSUB followed by the line number at which the subroutine starts. On completion of the allotted task, the keyword RETURN mysteriously directs control back to the calling program. In some respects, a subroutine behaves as a 'subcontractor' to the main program. And in much the same way that subcontractors will often be obliged to subcontract yet again, a subroutine may often call up another subroutine in order to complete a task.

This technique is called 'nesting' and may usually be extended to any level, subject to the limitations imposed by the area of memory known as the 'stack'. When GOSUB is used, the address of the next statement (which will be in an internal register called the Program Counter) is stored in the stack in order for the subroutine RETURN statement to know where to return to! If a subroutine calls on another, the appropriate address must again be stored.

If the level of nesting is increased too far, the poor old stack overflows and loses the last return address. The net result is either a system crash or at least an error message pointing out the memory deficiency. If an 'OUT OF MEMORY' error appears, it is not always due to actual shortage of memory — there may be several K left. The shortage is restricted to the relatively small allocation of stack memory. For efficiency purposes, the stack is normally located in Page zero or Page one of memory and is seldom more than a hundred or so bytes capacity in the majority of microcomputers. When this trouble arises, the only cure is to re-write some of the program to reduce the nesting level.

Originally, the term subroutine simply meant a block of coding intended to be used several times in the main program. If this block is written into the program every time it is needed, it is called an 'open loop' subroutine. If, on the other hand, it is written once only and used with GOSUB everytime, it is called a 'closed loop' subroutine. Although the normal closed loop method is universal and superficially more efficient in coding time and memory usage, a time penalty is involved. If execution time is paramount, an open loop block spliced in wherever it is needed will at least make a marginal improvement and may justify the extra bout of keyboard bashing. It should be pointed out that if speed is of the essence, the subroutine should be written in machine or assembly code anyway. However, this article is concerned with BASIC subroutines of the conventional closed loop type.

WHEN I'M CALLING YOU

Some subroutines can be called up straight away with GOSUB. Others may require a few preliminary assignment statements known as the 'calling sequence'. For example, a subroutine devoted to the simple task of drawing a screen line requires no calling sequence prior to GOSUB. On the other hand, a subroutine dedicated to perform some mathematical function on X and Y will require that the variables X and Y contain the numbers to be processed. If they currently reside in say, B3 and G2, then we must re-assign as follows before using GOSUB:

```
X=B3:Y=G2
GOSUB xxxx
```

These pre-assignments constitute the calling sequence.

On return from the subroutine, it may be necessary to apply the reverse procedure:

```
B3=X:G2=Y
```

The calling sequence in this example is a case of 'parameter-passing'. It may be argued with some justification that changing variable names should be unnecessary. Why couldn't the subroutine be written with variable names to suit the demands of the calling program? This is not always possible. For example, the same subroutine may be called up to perform the process at different times on different variables. The subroutine may have been written within a collection of general purpose subroutines in which case there could have been no prior knowledge of the variable names employed in subsequent programs.

WHAT KIND?

This question is similar to the 'length of the piece of string' caper. It is sufficient to remark that anything purporting to be ***general purpose*** cannot be ***specific*** in aim. The best attitude to adopt in choosing the list is to consider the task as an upgrading of the operating system and the BASIC interpreter. A high level language itself (such as BASIC) is in reality nothing more than a complex conglomerate of subroutines. These are, of course, written in machine code but nevertheless, they are still subroutines. Many of the facilities offered are taken for granted and we only take notice of the defects or omissions. Anyone who can write a high level interpreter or compiler would certainly have my respect.

In spite of this, for reasons of ROM space, any high level language cannot cover more than a sprinkling of keywords. A set of general purpose subroutines on tape or disc is equivalent to a language upgrading. Although no new keywords are added, the effect is the same. For example, most BASICs include SIN, COS, TAN and ATN but few offer the other inverse trig functions ASN (arc sine) and ACS (arc cosine). However, if GOSUB 5000 changes X into arc sine X then the BASIC is upgraded providing, of course, you have written such a subroutine and it is resident in memory. Naturally if your inclinations are such that arc sine X would only be used sparingly, if at all, then its inclusion would be foolish. General purpose subroutines are to some extent subjective to the individual.

Apart from augmenting the BASIC vocabulary, subroutines can be used to remedy defects in existing keywords. The INPUT statement in the PET has a truly diabolical feature. If the Return key is pressed (by accident) before the requested data has been keyed in, the program breaks out into COMMAND mode. A subroutine which replaces INPUT can easily be installed to remedy this effect.

Some subroutines can be very complex and others so simple as to be open to a charge of triviality. A simple subroutine, however, can still be a time and memory saver. For example, nothing could be more simple than a row of '—' across the screen in order to produce a line — but if many lines had to be drawn in the program, it would certainly justify a subroutine. On the other hand, it may be that a subroutine which requires an elaborate calling sequence each time might very well be counter productive. In such a case, it would be easier and more efficient to splice in the code each time it was used — an 'open loop' solution.

CHOICE OF VARIABLE NAMES . . .

It is good practice to choose variable names which suggest the data item (as far as it is possible with the two-character limit imposed by BASIC). When writing any form of general purpose subroutine it may be politic to ignore the rules in order to avoid clashes between the names. Unless care is taken, the subroutine may use working variables which are already used in the calling program with disastrous results. To act as a safeguard, a good plan is to use 'unusual' names in the subroutines in the belief that it would be too much of a coincidence to choose such names in the calling program. Using a high digit as the second character is also a good idea (thus N8 or K8).

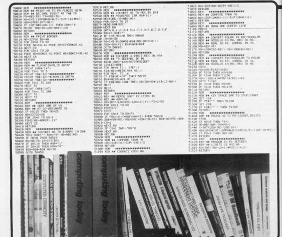
. . .AND LINE NUMBERS

If subroutines are placed at the head of the program with low line numbers there is, in most BASICs, a speed advantage because the GOSUB search commences from the top downwards. However, it is 'tidier' to place them down the bottom end. Where subroutines of the general purpose variety are concerned there is another reason why they should be at the bottom with high line numbers. BASICs which include the ability to APPEND programs together (or 'Toolkits') normally require the APPENDed code to be at higher line numbers than the already resident code. Thus, it is possible while developing a program to become aware of the need for the subroutines, in which case they can be appended.

USING GENERAL PURPOSE SUBROUTINES

A lazy way and one to which I am addicted, is to first load in the lot before beginning any program. If any are found to be useful then use them; any not used at the end can be erased.

After the program is more or less finished and tested, it may be possible to dispense with pre-assignments in the calling sequence by changing the ▶



```
50000 REM    ************************
50010 REM ** PRINT N8 TO P8 PLACES WITH
50020 REM ** DECIMAL POINT AT TAB T8
50030 P8=ABS(P8):T8=ABS(T8)
50040 N8=INT(10↑P8*N8+0.5)/INT((10↑P8))
50050 N8$=STR$(INT(N8))
50060 IF INT(ABS(N8))<1 THEN N8$="1"
50070 PRINT TAB(T8-LEN(N8$));N8
50080 RETURN
50090 REM    ************************
50100 REM ** PRINT BORDER
50110 A8=32768:B8=40
50120 FOR C8=0 TO 39
50130 POKE A8+C8,42:POKE A8+C8+B8*20,42
50140 NEXT C8
50150 FOR C8=1 TO 20
50160 POKE A8+B8*C8,42:POKE A8+B8*C8+39,42
50170 NEXT C8
50180 RETURN
50190 REM    ************************
50200 REM ** FLASH"INVALID DATA"
50210 PRINT CHR$(147)
50220 FOR T9=1 TO 5
50230 PRINT TAB(12)"***************"
50240 PRINT TAB(12)"*INVALID DATA*
50250 PRINT TAB(12)"***************"
50260 FOR T8=1 TO 200
50270 NEXT T8
50280 PRINT CHR$(147)
50290 FOR T8=1 TO 200
50300 NEXT T8
50310 NEXT T9
50320 RETURN
50330 REM    ************************
50340 REM ** VERT BAR OF N8
50350 REM ** AT CO-ORDINATE X8
50360 IF N8>25 THEN N8=25
50370 A8=33728+X8
50380 FOR Z8=0 TO N8-1
50390 POKE(A8-40*Z8),117
50400 NEXT Z8
50410 RETURN
50420 REM    ************************
50430 REM ** CONVERT N8 TO BINARY IN B0$
50440 T8=2:B0$="":R8$="0":N8=ABS(N8)
50450 IF N8=0 THEN 50510
50460 D8=N8/T8:I8=INT(D8)
50470 IF D8<>I8 THEN R8$="1"
50480 IF D8=I8 THEN R8$="0"
50490 B0$=R8$+B0$:N8=I8
50500 GOTO 50450
50510 RETURN
50520 REM    ************************
50530 REM ** CONVERT N8 TO HEX IN B8$
50540 REM ** REQUIRES DIM H8$(16)
50550 RESTORE:N8=ABS(N8)
50560 FOR E8=0 TO 15
50570 READ H8$(E8)
50580 NEXT E8
50590 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
50600 D8=16:B8$=""
50610 IF INT(N8)=0 THEN 50660
50620 N8=N8/D8
50630 R8=INT(0.00001+D8*(N8-INT(N8)))
50640 A8$=H8$(R8):B8$=A8$+B8$
50650 GOTO 50610
50660 RETURN
50670 REM    ************************
50680 REM ** CONVERT 4 DIGIT HEX IN A8$
50690 REM ** TO DECIMAL IN D8
50700 D8=0:H8$="123456789ABCDEF"
50710 L8=LEN(A8$)
50720 FOR A8=4 TO 1 STEP-1
50730 F8$(A8)=MID$(A8$,A8,1)
50740 FOR N8=1 TO 16
50750 IF F8$(A)="0" THEN 50790
50760 D8$=MID$(H8$,N8,1)
50770 IF F8$(A8)=D8$ THEN D8=D8+N8*(16↑(L8-A8))
50780 NEXT
50790 NEXT
50800 RETURN
50810 REM    ************************
50820 REM ** MERGE SORT E8 ITEMS IN
50830 REM ** A8$(N8)
50840 G8=INT((LOG(E8)/LOG(2))+1):F8=2↑G8
50850 FOR H8=1 TO G8
50860 F8=F8/2
50870 C8=0
50880 FOR N8=1 TO E8-F8
50890 IF A8$(N8)<=A8$(N8+F8) THEN 50920
50900 B8$=A8$(N8):A8$(N8)=A8$(N8+F8):A8$(N8+F8)=B8$
50910 C8=1
50920 NEXT N8
50930 IF C8=1 THEN 50870
50940 NEXT H8
50950 RETURN
50960 REM    ************************
50970 REM ** COMPUTE SINH N8
50980 N8=(EXP(N8)-EXP(-N8))/2
50990 RETURN
51000 REM    ************************
51010 REM ** COMPUTE COSH N8
51020 N8=(EXP(N8)+EXP(-N8))/2
51030 RETURN
51040 REM    ************************
51050 REM ** COMPUTE TANH N8
51060 N8=(EXP(2*N8)-1)/(EXP(2*N8)+1)
51070 RETURN
51080 REM    ************************
51090 REM ** DRAW ONE SCREEN LINE
51100 PRINT "----------------------------------------"
51110 RETURN
51120 REM    ************************
51130 REM ** CONVERT POLAR TO RECTANGULAR
51140 REM ** MODULUS IN M8, ARG IN A8,
51150 REM ** REAL IN R8, UNREAL IN X8.
51160 M8=ABS(M8)
51170 R8=(M8*COS(A8)):X8=(M8*SIN(A8))
51180 RETURN
51190 REM    ************************
51200 REM ** CONVERT RECTANGULAR TO POLAR
51210 REM ** REAL IN R8, UNREAL IN X8.
51220 REM ** MODULUS IN M8, ARG IN A8
51230 K8=0
51240 M8=SQR(R8↑2+X8↑2)
51250 IF R8<0 THEN K8=180
51260 IF R8=0 THEN 51290
51270 A8=((180/π)*ATN(X8/R8))+K8
51280 GOTO 51310
51290 IF X8>0 THEN A8=90
51300 IF X8<0 THEN A8=270
51310 RETURN
51320 REM    ************************
51330 REM ** KEY SPACE BAR TO STOP/START
51340 GET K8$
51350 IF K8$="" THEN 51380
51360 GET K8$
51370 IF K8$<>" " THEN 51360
51380 RETURN
51390 REM    ************************
51400 REM ** ROUND N8 TO P8 SIGNIF.DIGITS
51410 F8=0
51420 IF N8<0 THEN F8=1
51430 N8=ABS(N8):P8=P8-1
51440 C=INT(LOG(N8)/LOG(10))
51450 N8=10↑C*INT(10↑P8*N8/10↑C+0.5)/(INT(10↑P8))
51460 IF F8=1 THEN N8=-N8
51470 RETURN
51480 REM    ************************
51490 REM ** RANDOM IN R8 BETWEEN
51500 REM ** LIMITS L8 AND H8
51510 R8=INT((H8-L8+1)*RND(1)+L8)
51520 RETURN
```

subroutine variables to suit the calling program direct. It is wise to keep a copy of the pre-doctored version before attempting changes in the parameter names in case you land in a hopeless mess.

The listing shows the example set of subroutines which occupy lines 50000 to 51410. None of them are revolutionary, most of them lack elegance — but they do work on any PET and, with a few simple modifications, on most other computers. Line 50120 in the 'Print Border' subroutine initialises A8 to the starting address in the screen memory and B8 stores the number of characters in one screen line.

The POKE number 42 is the PET code for the character '*' which is used to paint the border. The random number subroutine at line 51390 may need a different keyword to RND(1). The 'Key Space Bar to STOP/START' subroutine at line 51250 may need changing if your GET waits for a character (the PET doesn't).

TESTING LOOPS

To avoid unnecessary clutter, the subroutines have a minimum of validation code. It is, of course, possible to build in traps to keep out absurd input parameters but the normal place of these would be after the INPUT statements in the calling program and it is inefficient to duplicate them again in the subroutine.

However, it is easy to try any of them out first by a simple loop which supplies the required parameters from INPUT. The following is a simple module to test out the first subroutine at line 50010:

```
100  INPUT "ENTER NUMBER TO BE
     PRINTED";N8
110  INPUT "ENTER NUMBER OF DEC
     PLACES REQUIRED";P8
120  INPUT "ENTER TAB POSITION OF
     DEC POINT";T8
130  GOSUB 50010
140  GOTO 100
```

The method of testing out the others would be similar in principle. The GOTO 100 allows you to try out all possible inputs until you are satisfied with the extent of the limitations.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Pieter Hintjens

TOOLKIT PROGRAM

Here is a fairly comprehensive machine toolkit program for use on a Gommodore 64

When programming your Commodore 64, you'll soon want features like function key assignment, search and others available only on an expensive toolkit expansion. Until now, because this program gives you a fairly comprehensive machine code toolkit that offers features including all the above and more.

Briefly, for the small price of 1k of memory and a few hours programming, you can:

— assign up to 32 characters to the eight function keys F1 to F8

— search a program for a string, printing out any 'finds'

— step through screen and border colour combinations at will

— stop the execution of any program so that you can inspect the screen, then continue running it as if nothing had happened

— have a bell ring whenever your typing nears the end of the line.

When in operation, the toolkit becomes part of the system IRQ routine which is called 60 times every second to update the clock, see if the stop key is pressed and so on. When called, the toolkit can check whether a certain key is pressed and if so, do some action. For example, if some character string has been assigned to the key F1 and the toolkit sees that F1 is pressed, it will output that string.

THINGS TO KNOW

Before you can use the toolkit, you must enter and check the BASIC loader program. The machine code is contained in the data statements in lines 100 to 470. While entering the loader, which is quite large, you should make frequent backup copies so that if a jealous pet trips the wire, you don't have to start all over again. Check the program and have a copy saved before you run it.

When run, the program loads data into memory, checking all the time for errors in the data statements. Should it spot one, it will tell you the line number and abort the load. You should find and correct the error and then try again. There is also a check made on the overall sum of the data, so that an error not found by the checksum at the end of every data line may still be spotted. However, in this case you will not be told the line number.

When the toolkit is initialised (the loader does this for you but the way to do it is to type 'SYS 156*256 (return)') the message '** CBM 64 TOOLKIT (c) 1983 P. HINTJENS **' will be printed. The first time the toolkit is initialised, the function key assignments are cleared and the bell tab reset. However, after that it can be switched on and off and the function key assignments will not be wiped.

The first function provided is a 'bell tab'. This means that while you are typing a line, a bell will be rung when you reach the 75th character. The reason for this, apart from knowing where you are is that the CBM 64 (mine at least) has a ROM bug which crashes rather messily if you over-run the last screen line and try to delete your way back up. If you do this, the computer is lost and has to be switched off.

IN COMMAND

The second function provided is the access to eight special commands using the Ctrl key in conjunction with one of the eight function keys (shifted and unshifted). These commands are:

F1 — FIND — Using the first program line as the search string, the program in memory is searched and any occurrences of the data are listed on the screen. After a line is listed, the toolkit waits for you to press a key. If you press Stop, the search is cancelled. If you press L, the toolkit starts listing from the point of the last 'find'. If any other key is pressed, the search goes on. When the L option is chosen, you also press a key for the next line or Stop to stop. To use this facility, you enter the data to be looked for as the first program line (I keep line 0 for this purpose) and then press Ctrl and F1.

F2 — ASSIGN — This facility allows you to assign up to 32 characters to one of the eight function keys. Pressing Ctrl, Shift and F1 together will display a 'F' followed by a cursor. Enter the number (1 to 8) of the key and a space, followed by the command string. In some cases it can be useful to incorporate a carriage-return character into the command string — to do this, the '←' character should be entered. For example: (the computer's output is underlined):

F7 LIST THIS LINE ← (return)

Now whenever the key F7 is pressed, 'LIST THIS LINE (return)' will be printed. Special provision is made so that if you are in quotes mode (when the function keys produce those strange characters) no command string will be output. ▶

F3 — PAUSE — It is often useful to be able to stop a program's execution for a time so that you can inspect the screen and then continue, for example during a list. Pressing Ctrl and F3 will do this; pressing them again will continue whatever was being done. While the toolkit is waiting for you to press the keys again, you may use the Stop key — a single push will do the same as the Ctrl-F3 combination; however, if you hold it down, a 'break' will be caused. Again this is useful during a list.

F4 — KILL — If you should need to disable the toolkit, use this command. The alternative is Stop-Restore, but this will reset the whole system, screen colours and all by way of a warm start. A message will tell you that all is well and the toolkit will no longer function. By initialising it (SYS 156 *256) it will come back to life, though any assigned command strings will be kept as they were.

F5 — TAB MOVE — It may be desirable to move the tab position along the line to suit a taste or purpose; this command will move it one position to the right; after 79 it is moved back to 65.

F6 — TAB DISPLAY — To see where the tab position is, other than moving the cursor along until you hear a ring, use this command. A line of dots will be printed up to the tab position.

F7 — SCREEN COLOUR — Rather than continually POKEing some obscure location (I know it's 53281 and you may know that, but don't spoil this), this command will step through the 16 available screen colours, starting at 0 (black) when it goes past 15.

F8 — BORDER COLOUR — This does the same for the border colour. The beauty of this command and Screen Colour is that you can set the colours while a program is running.

The toolkit program, in its machine code form uses the top 1k of memory from BASIC, starting at $9C00 up to $9F00 — a 256 byte chunk is used to store the function key assignments. The free 4k block from $C000 to $D000 was avoided, because every other machine code program will use it. Once you have a working copy of the toolkit, it would be better and more efficient to make a saved copy of the actual machine code rather than the indirect BASIC loader. To do this, you fool the 64 into thinking that there is a BASIC program starting at $9C00 and ending at $9F00 (156 *256 to 159 *256):

```
POKE 44,156:POKE 43,0
POKE 46,159:POKE 45,0
SAVE "TOOLKIT"
```

Then, to load the machine code directly into the same area of memory, use:

```
LOAD "TOOLKIT",1,1
POKE 56,156:POKE 52,156:NEW
SYS 156*256
```

```
100 DATA 120,169,41,141,20,3,169,156,141,21,3,32,161,158,173,157, 129
110 DATA 2,208,20,238,157,2,162,0,142,155,2,169,75,141,156,2, 95
120 DATA 138,157,0,159,232,208,250,88,96,72,152,72,138,72,165,204, 155
130 DATA 72,169,1,133,204,166,211,173,155,2,208,13,236,156,2,208, 61
140 DATA 16,32,244,157,238,155,2,208,8,236,156,2,240,3,206,155, 10
150 DATA 2,165,197,197,251,240,30,133,251,201,3,48,24,201,7,16, 174
160 DATA 20,170,169,0,133,207,164,211,165,206,145,209,173,141,2,74, 141
170 DATA 240,14,74,208,56,104,133,204,104,170,104,168,104,76,49,234, 250
180 DATA 165,212,208,241,32,24,158,10,10,10,10,10,170,160,32,189, 105
190 DATA 0,159,240,22,201,95,208,11,169,13,141,119,2,169,1,133, 147
200 DATA 198,208,3,32,210,255,232,136,208,229,76,117,156,32,24,158, 226
210 DATA 10,170,189,191,156,133,252,189,192,156,133,253,108,252,0,14, 94
220 DATA 157,10,157,27,157,79,158,223,156,207,156,39,158,55,158,32, 137
230 DATA 175,158,169,49,141,20,3,169,234,141,21,3,76,117,156,32, 128
240 DATA 159,255,165,197,201,64,208,247,32,159,255,165,197,174,141,2, 61
250 DATA 201,63,240,11,201,5,208,240,224,4,208,236,76,117,156,169, 55
260 DATA 3,141,119,2,169,1,133,198,208,242,162,0,240,2,162,1, 247
270 DATA 188,32,208,200,152,157,32,208,76,117,156,165,43,133,252,165, 236
280 DATA 44,133,253,24,144,14,160,3,200,177,252,205,5,8,240,12, 82
290 DATA 201,0,208,244,32,226,157,208,237,76,117,156,132,254,162,0, 106
300 DATA 232,200,189,5,8,201,0,240,9,209,252,240,243,164,254,24, 166
310 DATA 144,214,32,121,157,32,228,255,201,0,240,249,201,3,240,217, 230
320 DATA 201,76,208,208,32,226,157,240,208,32,121,157,32,228,255,201, 22
330 DATA 0,240,249,201,3,208,237,240,192,169,18,32,210,255,160,2, 112
340 DATA 177,252,170,200,177,252,32,205,189,160,3,169,146,32,210,255, 69
350 DATA 169,32,32,210,255,200,177,252,240,52,201,255,240,42,201,34, 32
360 DATA 240,50,201,128,144,34,56,233,127,170,132,254,160,255,202,240, 66
370 DATA 8,200,185,158,160,16,250,48,245,200,185,158,160,48,5,32, 10
380 DATA 210,255,208,245,41,127,164,254,32,210,255,24,144,199,169,13, 246
390 DATA 32,210,255,96,32,210,255,200,177,252,240,242,201,34,208,244, 72
400 DATA 240,230,160,0,177,252,170,200,177,252,134,252,133,253,177,252, 243
410 DATA 136,17,252,96,169,0,141,0,212,169,64,141,1,212,169,37, 24
420 DATA 141,5,212,169,5,141,6,212,169,15,141,24,212,169,17,141, 243
430 DATA 4,212,169,32,141,4,212,96,173,141,2,41,1,133,254,138, 217
440 DATA 56,233,3,10,5,254,96,174,156,2,232,224,80,208,2,162, 105
450 DATA 60,142,156,2,76,117,156,169,13,32,210,255,174,156,2,169, 97
460 DATA 46,32,210,255,202,16,250,169,13,32,210,255,76,117,156,169, 160
470 DATA 13,32,210,255,169,70,32,210,255,162,0,32,207,255,157,60, 71
480 DATA 3,232,201,13,208,245,32,210,255,202,169,0,157,60,3,173, 115
490 DATA 60,3,56,233,49,41,7,170,189,153,158,10,10,10,10,10, 145
500 DATA 133,252,169,159,133,253,160,0,185,62,3,145,252,201,0,240, 43
510 DATA 5,200,192,32,208,242,76,117,156,2,3,4,5,6,7,0, 231
520 DATA 1,162,0,189,188,158,240,6,32,210,255,232,208,245,96,162, 80
530 DATA 0,189,232,158,240,248,32,210,255,232,208,245,13,13,13,42, 26
540 DATA 42,32,67,66,77,45,54,52,32,84,79,79,76,75,73,84, 249
550 DATA 32,40,67,41,32,49,57,56,51,32,80,46,72,73,78,84, 122
560 DATA 74,69,78,83,32,42,42,0,13,66,89,69,45,66,89,69, 158
570 DATA 46,13,0,0,255,255,0,0,255,255,0,0,255,255,0,0, 53
1000 POKE56,156:POKE52,156:REM ** MACHINE CODE STARTS AT $9C00
1005 CLR
1010 PRINT "[CLS]** TOOLKIT LOADER **"
1020 PRINT "LOADING AT 39936: ";
1030 ALLCHECK=0:REM ** KEEPS A COUNT FOR ALL DATA
1040 CODE=156*256:REM ** START OF CODE
1050 FOR LINE=0 TO 47:REM ** 48 DATA LINES
1060 CHECK=0:REM ** KEEPS A COUNT FOR THE LINE
1070 FOR BYTE=0 TO 15:REM ** 16 DATA ITEMS PER LINE
1080 READ ITEM
1090 CHECK=CHECK+ITEM
1100 POKE CODE+LINE*16+BYTE,ITEM
1110 NEXT BYTE
1120 READ SUM:REM ** CHECKSUM
1130 IF SUM<>(CHECK AND255) THEN 2000:REM ** CHECK SUM ERROR IN LINE
1140 ALLCHECK=ALLCHECK+SUM+CHECK
1150 PRINT ".";
1160 NEXT LINE
1170 IF ALLCHECK<>104764 THEN 2100:REM ** DATA SHOULD ADD UP TO 104764
1180 SYS 156*256:REM ** INITIALISE TOOLKIT
1181 END
2000 PRINT "[CD]!DATA ERROR IN LINE";PEEK(63)+256*PEEK(64)
2010 PRINT "LOAD ABORTED."
2011 END
2100 PRINT "[CD]!UNLOCATED DATA ERROR"
2101 GOTO 2010
```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Chris Wyatt

TAILORING VIC'S CHARACTERS

Here is a utility program that allows you to create your own symbols on the VIC.

The VIC-20 is provided with an excellent set of graphics characters stored within its ROM. But like all the best of today's micros the VIC is not limited to just these 'off-the-peg' characters. The character generator on the VIC can be redefined, enabling it to produce a virtually limitless set of 'user-created' character shapes, like 'proper' Space Invaders, Lunar landers, tanks, Pacmen(?) and even people.

Whilst the creation of user-defined characters is not especially difficult, the processes involved can be extremely tedious, usually requiring copious quantities of squared graph paper and furious thumbing of pocket-calculator buttons.

This program provides you with a high-resolution, on-screen character editor, together with automatic calculation of the POKE values of the new character for incorporating into other programs. Furthermore, a shape defined within the boundaries of just one character position is bound to be somewhat limited. The program is therefore arranged to work on a 2 by 2 character matrix, characters being formed within 1, 2, 3, or 4 character positions.

EIGHT TO GENERATE

But let us first quickly recap on how the VIC's character generator works. Each character that the VIC can PRINT can be considered as being made up of eight horizontal rows of eight pixels per row. Each row of pixels is defined in a separate address in memory, which contains the information as to which of the eight pixels in the row will be 'set', and which will be 'not set'.

These 'row-addresses' will contain a binary number with a value between 00000000 and 11111111, ie between 0 and 255. The pattern of 1s and 0s that this binary number represents is directly related to the pattern of pixels 'set' and 'not set' in the row. This is illustrated in Fig. 1.

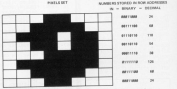


Fig. 1. The correspondence between pixel sets and binary numbers stored in row addresses is illustrated.

Since each full character is made up of eight rows of pixels, the complete character shape can be defined by the numbers stored in eight consecutive row-addresses. The block of addresses used for holding the shape information for the whole character set is normally referred to as the 'character generator'. On the VIC-20 this is located in ROM starting at address 32768. Thus, the POKE code 0 (an '@' symbol) uses the information in addresses 32768 to 32775; code 1 (an 'A' uses) addresses 32776 to 32783, and so on.

However, there is essentially no difference between an address containing character shape information and any other address. All are eight bits wide, all will contain numbers between 0 and 255, so any of them could be used to represent one row of pixels. We cannot change the ROM character generator itself, but if we can tell the VIC to refer to a new series of addresses for its character shape information — an area in RAM which we can change — then we have the ability to define our own character shapes.

Address 36869 is used (amongst other things) as a pointer to the start of the character generator. Normally it contains 240, which is used to point to address 32768. If we change the contents of 36869 we can redirect the VIC to a new character generator start address. For example, POKE 36869,255 will point to a character generator starting at address 7168, a RAM address.

This explanation of the VIC character generator is somewhat over simplified so purists can accept my apologies now. It is nevertheless accurate as far as it goes — more detailed explanations can be found in the **VIC Programmers Reference Guide.**

Having got the VIC to look somewhere else for its character generator all POKE statements to the screen will reference the new row-addresses.

That is:

POKE (any screen address), 0

will use the information in the ▶

first eight addresses of the new character generator. Strangely enough, the VIC will still regard character code 0 as an '@' symbol, code 1 as an 'A', etc, so the statement PRINT "@" is still perfectly valid even though the character that appears on the screen looks nothing like an '@'.

The VIC will merely assume that by '@' we mean the contents of the first eight addresses of the character generator, regardless of where in memory the character generator is located.

THE PROGRAM

The program is written in BASIC for the unexpanded VIC-20, and is, as previously stated a 'character editor'.

The main screen display consists of a 16 by 16 'grid' of full size character positions at normal resolution. The grid represents the rows of pixels forming four full characters arranged in the formation:

0	1
2	3

A flashing cursor can be moved around the grid using the normal cursor control keys, and individual elements of the grid can be 'set' using function key f1 or 'unset' using function key f3. This setting or unsetting of elements at normal resolution is reflected at high resolution in an area immediately below the grid. So for every grid element which is set, a single pixel will appear at the bottom of the screen in exactly the same relative location.

Thus, as you move around the main display, turning elements on and off, your actions are mirrored in high resolution — showing you exactly how your finished character will look.

Included in the program are 'paint' and 'unpaint' routines. Paint, called by function key f2, will begin setting each pixel following the current cursor position until either you stop it by pressing any key, or it fills in the whole area remaining. Unpaint — function key f4 —

```
 10  POKE 56,28:REM ** LOWER RAMTOP TO PROTECT NEW
     CHARACTER GENERATOR
 20  DATA 1,1,7,15,7,3,1,1:REM ** FIRST CHARACTER
 30  DATA 64,160,0,192,224,224,224,224:REM ** SECOND
 40  DATA 3,135,207,241,127,106,63,63:REM ** THIRD
 50  DATA 199,134,14,62,252,172,248,240:REM ** FOURTH
 60  DATA 999:REM ** END OF LIST POINTER
 70  RESTORE
 80  I=0
 90  READ A:IF A=999 THEN 110
100  POKE 7168+I,A:I=I+1:GOTO 90
110  POKE 36869,255:REM ** POINT TO NEW CHARACTER
     GENERATOR AT 7168
120  REST OF PROGRAM ...
```

Fig. 5. A program showing one method of incorporating character shape values.

```
  9  REM ** INITIALISE NEW CHARACTER GENERATOR
 10  POKE 56,28:FOR I=7168+32 TO 7168+39:POKE I,255:NEXT
 20  FOR I=7168 TO 7168+4*8-1:POKE I,0:NEXT
 30  PRINT "[CLS]";:POKE 36869,255
 39  REM ** MAIN DISPLAY
 40  PRINT "[REV][3 SPC]7654321076543210"
 50  PRINT "[REV][3 SPC][16 G<@]"
 60  FOR Z=1 TO 2
 70  FOR I=1 TO 8:PRINT "[REV]";I;"[CL][G<M][16 SPC]
     [G<G]":NEXT:NEXT
 80  PRINT "[REV][3 SPC][16 G<T][OFF]"
 89  REM ** INITIALISE MAIN VARIABLES
 90  TL=7727:CL=38447:PP=TL:CP=CL:X=1:Y=1:MX=128:DL=100
 99  REM ** PRINT NEW CHARACTERS
100  PRINT "[HOM][19 CD][9 CR]@A[CD][2 CL]BC"
109  REM ** KEYBOARD INPUT
110  GET A$:IF A$="" THEN AP=0:GOSUB 500:GOTO 110
120  AP=0
130  IF A$="[CD]" THEN GOSUB 230:GOSUB 500:GOTO 110
140  IF A$="[CU]" THEN GOSUB 260:GOSUB 500:GOTO 110
150  IF A$="[CR]" THEN GOSUB 290:GOSUB 500:GOTO 110
160  IF A$="[CL]" THEN GOSUB 330:GOSUB 500:GOTO 110
170  IF A$="[F1]" THEN GOSUB 370:GOTO 110
180  IF A$="[F2]" THEN GOSUB 710:GOTO 90
190  IF A$="[F3]" THEN GOSUB 440:GOTO 110
200  IF A$="[F4]" THEN GOSUB 900:GOTO 90
210  IF A$="[F8]" THEN GOSUB 580
220  GOTO 110
229  REM ** MOVE DOWN
230  IF Y>15 THEN RETURN
240  Y=Y+1:AP=22
250  RETURN
259  REM ** MOVE UP
260  IF Y<2 THEN RETURN
270  Y=Y-1:AP=-22
280  RETURN
289  REM ** MOVE RIGHT
290  IF X>15 THEN RETURN
300  X=X+1:AP=1
310  MX=MX/2:IF MX<1 THEN MX=128
320  RETURN
329  REM ** MOVE LEFT
330  IF X<2 THEN RETURN
340  X=X-1:AP=-1
350  MX=MX*2:IF MX>128 THEN MX=1
360  RETURN
369  REM ** SET PIXEL
370  CB=(INT(Y/9)*2)+INT(X/9)
380  IF Y>8 THEN PY=Y-8:GOTO 400
390  PY=Y
400  CV=7168+(CB*8)+PY-1
410  POKE PP,4:POKE CP,0
420  POKE CV,PEEK(CV) OR MX
430  RETURN
439  REM ** UNSET PIXEL
440  CM=(INT(Y/9)*2)+INT(X/9)
450  IF Y>8 THEN PY=Y-8:GOTO 470
```

```
460  PY=Y
470  CV=7168+(CM*8)+PY-1
480  POKE PP,160:POKE CV,PEEK(CV)-(PEEK(CV) AND MX)
490  RETURN
499  REM ** FLASHING CURSOR
500  PP=PP+AP:CP=CP+AP
510  RM=PEEK(PP):CM=PEEK(CP)
520  POKE PP,4:POKE CP,5
530  FOR I=1 TO DL:NEXT I
540  POKE PP,RM:POKE CP,CM
550  FOR I=1 TO DL:NEXT I
560  AP=0
570  RETURN
579  REM ** OUTPUT CHARACTER DATA
580  PRINT "[HOM][22 CD][REV]CONFIRM-ANY KEY![OFF]";
590  CF=0
600  CF=CF+1:IF CF>300 THEN PRINT "[HOM][22 CD][REV]
     [19 SPC][OFF]";:RETURN
610  GET CF$:IF CF$="" THEN 600
620  PRINT "[CLS]"
630  FOR I=0 TO 7:PRINT "[REV]";TAB(2);PEEK(7168+I);
     TAB(14);PEEK(7168+I+8):NEXT:PRINT
640  FOR I=16 TO 23:PRINT "[REV]";TAB(2);PEEK(7168+I);
     TAB(14);PEEK(7168+I+8):NEXT:PRINT
650  PRINT "[HOM][4 CD][CR]@[11 CR]A"
660  PRINT "[HOM][13 CD][CR]B[11 CR]C"
670  PRINT "[5 CD][9 CR]@A[CD][2 CL]BC"
680  PRINT "[CD][REV]TYPE 'C' TO CLEAR";
690  GET CC$:IF CC$="" THEN 690
700  PRINT "[CLS]":POKE 36869,240:END
709  REM ** PAINT
710  DL=10
720  PRINT "[HOM][22 CD][REV]ANY KEY TO STOP[OFF]";
730  FOR YY=1 TO 8
740  FOR XX=1 TO 16
750  GOSUB 370:GOSUB 290:GOSUB 500
760  GET Z$:IF Z$="" THEN 780
770  GOTO 870
780  NEXT XX
790  GOSUB 230:GOSUB 500
800  FOR XX=1 TO 16
810  GOSUB 370:GOSUB 330:GOSUB 500
820  GET Z$:IF Z$="" THEN 840
830  GOTO 870
840  NEXT XX
850  GOSUB 230:GOSUB 500
860  NEXT YY
870  DL=100
880  PRINT "[HOM][22 CD][REV][17 SPC][OFF]";
890  RETURN
899  REM ** UNPAINT
900  DL=10
910  PRINT "[HOM][22 CD][REV]ANY KEY TO STOP[OFF]";
920  FOR YY=1 TO 8
930  FOR XX=1 TO 16
940  GOSUB 440:GOSUB 290:GOSUB 500
950  GET Z$:IF Z$="" THEN 970
960  GOTO 1060
970  NEXT XX
980  GOSUB 230:GOSUB 500
990  FOR XX=1 TO 16
1000 GOSUB 440:GOSUB 330:GOSUB 500
1010 GET Z$:IF Z$="" THEN 1030
1020 GOTO 1060
1030 NEXT XX
1040 GOSUB 230:GOSUB 500
1050 NEXT YY
1060 DL=100
1070 PRINT "[HOM][22 CD][REV][17 SPC][OFF]";
1080 RETURN
```

Listing 1. A program for character generation on the unexpanded VIC-20.

does the reverse, ie it wipes out any pixel already set.

The high resolution character at the bottom of the screen is, in fact, a composite of the first four characters in a new character generator, starting at address 7168. These characters — character POKE codes 0, 1, 2, and 3 or, if you wish, PRINT symbols @, A, B, and C — are arranged on the screen in the same formation as the main grid. Editing the grid causes corresponding changes in the contents of the row-addresses of these characters, and hence changes in their shape.

When you are satisfied with your new character, a read-out of the values now stored in these 32 new row-addresses can be called via function key f8, but since displaying these values will effectively destroy the main grid display, two safety mechanisms are built in. First, the key f8 can only be operated using the SHIFT key at the same time, ie f8 is Shifted f7, so you can't press it accidentally. Secondly, even after f8 has been pressed, the program requires you to 'confirm' your request. If you do nothing, the program will, after a short pause, revert to the normal editing mode with the display intact.

Assuming that you have actually finished editing, and now require a readout of the character shape values, the values will be displayed on the screen in four blocks, each block corresponding to one character in the 2 by 2 matrix. Alongside each block of eight numbers the new character shape formed by them is displayed, and, at the bottom of the screen, the full 2 by 2 character is repeated.

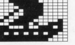


Fig. 2. RN Long range.

All you have to do now is to record the values displayed for inclusion in your object program. Remember, however, the relative position of each of the individual characters within the 2 by 2 matrix.

ie $\begin{matrix}0 & 1\\ 2 & 3\end{matrix}$ or $\begin{matrix}@ & A\\ B & C\end{matrix}$

This must be preserved in your object program.

One method of incorporating the character shape values into a new program is shown below. In this example the shape values of a 2 by 2 character are POKEd into the first 32 addresses of a new character generator starting at 7168. Generally you will want a variety of different characters in your program, so the use of a counter, I, together with a dummy DATA value at the end of the DATA list, 999, enables you to include extra characters as required without having to worry about the controlling variable in a FOR . . . NEXT loop.

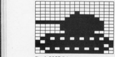


Fig. 3. BAOR firing range.

LISTING NOTES

The listing given is for the unexpanded VIC-20, and uses the standard ***Computing Today*** conventions throughout.

The program could have been compacted quite a lot with further use of subroutines but has been left 'open' for the sake of clarity.

For those of you who don't have a VIC, the following notes should enable the program to be converted to other machines with a similarly organized character generator.

POKE 56,28 lowers the top of RAM to protect a character generator starting at 7168.

The VIC's screen display comprises 23 rows of 22 character positions per row, giving 506 positions in all. The screen is mapped into two separate areas of memory:

1) 7680 to 8185 is used to hold the code for each character displayed.
2) 38400 to 38905 hold the colour code for each character displayed. (I'm oversimplifying again but it's close enough)

Thus, POKEing any character onto the screen requires two separate instructions:

POKE 7680,1 : POKE 38400,0

causes a black letter 'A' to be displayed at the top left corner of the screen, 1 being the POKE code for 'A', and 0 being the colour code for Black.

The characters shown as printed in Reverse video in the listing do not actually appear in their reverse form. Without going into details, this particular location of character generator enables normal resolution and high resolution character (Yes, I know they're both high resolution really) to be mixed on the same screen without copying from the ROM character generator into the new one. You simply PRINT the normal characters in their reversed form — handy eh?

The graphics characters listed in standard ***Computing Today*** format are:

Line 50 G @ is a horizontal bottom bar. (▁)
Line 70 G M is a vertical right-hand bar. (▕)
G G is a vertical left-hand bar. (▏)
Line 80 G T is a horizontal top bar. (▔)

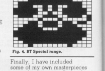


Fig. 4. BT Special range.

Finally, I have included some of my own masterpieces using the program. Sorry about the rather feeble joke in the titles — I just wanted to give British Telecom's Special Range phones a plug, (PUN).

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

TIRED OF TYPING?

Why type in thousands of bytes of BASIC when you can buy all these programs ready to LOAD?

Micro Examination
Test your friends and children with this multiple choice program.

Multicolumn Records
Set up your own filing system which enables you to store, search, edit and retrieve data.

Multipurpose Records
A multipurpose data base program for use at home or in the office.

Quiz Time
Assess your performance in terms of speed and accuracy with a multiple choice program.

Communications
Get Commodores talking to each other using this application.

Address Book
Compile an address book, or any other similar list, and throw away those bits of paper.

VIC Blow Up
Find out how characters are made up and generate giant versions on the VIC-20.

Toolkit Program
A simple toolkit program for the Commodore 64.

Tailoring VIC's Character
Create your own characters on your VIC-20.

VIC Editor
Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.

CBM TAPE 1 ☐

CBM TAPE 2 ☐

Tapes are priced at £9.99 each (all inclusive) or you can order two tapes for £15.99

ASP Software
ASP Ltd
145 Charing Cross Road
London WC2H 0EE

I enclose my Cheque/Postal Order/International Money Order for: (delete as necessary)
£. . . . .(made payable to ASP Ltd)
OR debit my Access/Barclaycard
(delete as necessary)

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

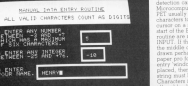
Please use BLOCK CAPITALS
Name (Mr/ Mrs/ Miss) . . . .
Address . . . .
. . . .**Postcode** . . . .
Signature . . . . **Date** . . . .

Please allow 21 days for delivery

R E Wellings Thomas

MAXI-MANDER

Operator error can cause even the best program to crash. We show you how to prevent most of the common mistakes getting through.



```
          MANUAL DATA ENTRY ROUTINE
NB. ALL VALID CHARACTERS COUNT AS DIGITS

** ENTER ANY NUMBER
 BETWEEN  -3 AND  +7
 WHICH HAS A MAXIMUM         5
 OF SIX CHARACTERS.

** ENTER ANY INTEGER
 BETWEEN -25 AND +76.        -10

** ENTER
 YOUR NAME.     HENRY█
```

The recent rapid proliferation of microcomputers has meant that more and more people totally untrained in their use will find themselves trying to communicate with, or through a machine. Such applications, already clearly established in computer-aided learning (CAL), computer-assisted medical diagnosis, or computerised help for the handicapped, will expand rapidly. What about computer-aided bank loans, insurance, travel, or even gardening? All of these situations rely on an intelligent dialogue between the computer and the user. Current technology usually demands that the user must reply to a number of program-generated questions. A reply is normally effected by typing an answer. It is the Manual Data Entry process, the 'moment of truth' in computer-aided transactions, that is the subject of this article.

STRATEGIES

There are two complementary strategies for data entry. The first takes a data string, whether numerical or alphabetical, and then tests it to check whether that particular input conforms to program requirements before accepting it. The second strategy takes each character, one at a time, as it is entered, checks it, and then concatenates it into a data string. BASIC supports both strategies and implements the first as the standard string INPUT statement. It is clear that this input data may be separated character by character and subsequently tested using string analysis techniques. An example is given below:

```
Statement               Function

10  INPUT A$            Get an input
20  AL=LEN(A$)          Find its length
30  FOR C=1 TO AL
40  B$=RIGHT$(A$,C)     Test each
                        right-hand
50  :  :                character for
                        some
60  NEXT C              characteristic
                        etc
```

Line 50 would normally check to see if the ASCII value of the character fell into some acceptable range, possibly numerical. In the latter case a letter would generate an error.

It is self-evident that, in using the INPUT strategy, the whole of the data string has had to be entered before error detection can begin. Microcomputers such as the PET usually require that all characters to the right of the cursor on a screen line at the start of the BASIC string INPUT routine are read as part of the INPUT. If the INPUT occurs in the middle of a graphic display, drawn perhaps to simulate a paper pro forma in which data entry 'windows' have been placed, then the length of the string must be checked. Characters in excess of the allowable character length must be deleted and the mutilated screen pro forma redrawn. In the example below the BASIC routine previously described has been modified to include such a routine.

```
Statement                  Function

 35  IF AL>IL THEN 70      IL=selected
                           input length
 :   :          :
 70  FOR C=AL TO IL
     STEP-1
 80  PRINT "[CL][SPC]      Delete extra
     [CL]"                 character

 90  NEXT C
100  PRINT "......"        Reprint
                           proforma
                           window
```

Line 80 merely replaces the PET Cursor control characters with printable ones.

BETTER STRATEGY

The second 'character input' strategy is the better strategy because it enables remedial action to be taken as soon as a false input character is detected. If can be implemented in BASIC using the GET statement. The GET is performed almost instantaneously and will return a zero, or "", indicating a null string, even if no key is pressed. During GET no cursor or characters are displayed and the input string is limited to one character. This means that a lengthy input string will have to be concatenated from a succession of characters before use. The ability to process individual characters before acceptance, despite cursor, keyboard and display problems, makes the use of the GET statement very attractive for properly validated data entry routines.

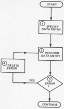


Fig. 1. The simplified data input routine.

DATA ENTRY REQUIREMENTS

It is always good practice to think before acting. It is equally good programming practice to begin with an algorithm, often expressed in a flow chart, however simple. Let us begin by considering the algorithm given in Fig. 1. At least it is simple; it clearly indicates the order of events and divides the task into four separate parts.

The first box of the algorithm in Fig. 1 asked for the specification of the type of data entry required. Let us specify four types. Obviously further types could be added but the following input types (IT) should suit most applications.

IT = 0 allows any character or symbol
IT = 1 allows numbers only
IT = 2 allows integers only
IT = 3 allows letters only

Numerical input is often required to fall between set upper and lower limits and it would be appropriate to specify these limits here. If the data string has to fit inside a particular graphics 'window', then this input length (IL) should also be specified at this point. A typical specification might read as follows:

IT = 2
IL = 2
UL = 40
LL = −2

This would mean a numerical integer input, two characters wide, having an upper limit (UL) of 40 and a lower limit (LL) of −2.

ENTRY ROUTINE

Let us now develop a data entry routine based on the preferred 'character input' strategy using the BASIC GET statement. Fig. 2 shows a possible algorithm.

The first point to notice in Fig. 2 is that a decision box is used to show the detection of null values of the input character C$, which then loops control back to the GET box. Should C$ get the 'RETURN' charcter. ASCII Code (13), then there must be a suitable method of ending the input.

A second decision box therefore shows the detection of CHR$(13) and passes control to the end of the routine. The specification of the input is now relatively complex and a number of different error conditions could occur. It would be useful to tell an unsuccessful user why his data entry has failed to be accepted. Suitable error messsages will need to be generated showing how the specified input conditions have been contravened. Notice that the only entry to the error handling routines is through the setting of an error flag. If the error flag (EF) is made to take more than the usual binary states (set and clear) then the flag itself will trigger the appropriate response in the error handling subroutines.

A second method of terminating the input routine would be for the actual input string length (AL) to equal the specified input length (IL). This technique avoids distrubing the display graphics. The PRINT statement is required because the GET statement, unlike INPUT, does ***not*** display characters as they are keyed in.

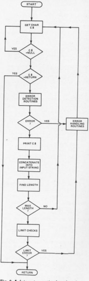


Fig. 2. A data entry routine based on the GET instruction.

ERROR DETECTION

The actual checking of each character (C$) as it is entered is ▶

relatively straightforward once the input type variable (IT) has been specified. The flow chart in Fig. 3 shows a suitable algorithm.

First of all the ASCII code value of the character must be determined, and assigned as the value of an input variable (IV). Then the value contained in the IT variable must be used to direct program control to that area of code where IV can be compared with the ASCII codes of unacceptable characters. If a match is found then an error flag (EF) can be set. Code comparisons can be made using BASIC condition statements of the IF . . . THEN variety suitably combined with AND and OR operators. This in turn will assign a value representing the error to the error flag variable (EF), whenever the condition statements evaluate to TRUE.

If, on the other hand, the condition statements evaluate to FALSE, then the input character (C$) will be concatenated into the string (I$). If the IT value represents numerical input the IV must be reassigned the numerical value of I$ after which limit checks can be made.

The routine below shows a much simplified version of the technique. The numerical input given in line 200 would in reality need to be expanded to include the decimal point () as well as the plus (+) and the minus (−) signs as valid characters.

```
Statement                   Function

100  GET C$                 Get a
                            character
110  IF C$="" THEN 100      Loop on C$=
                            null string
120  IV=ASC(C$)             Input Value
                            (IV)=ASCII
                            value of
                            input
130  IF IT=1 THEN XXX       This section
                            directs
                            different
140  IF IT=2 THEN 200       values of
                            Input Type
                            (IT) to
150  IF IT=3 THEN YYY       relevant
:    :        :             condition
                            statements
200  IF IV<48 OR IV>57      This only
     THEN EF=3              allows numbers
                            as valid
                            characters
```

The error flag variable (EF) can be clearly seen to take different values dependent on the input type. Suitable values of EF are given below together with an appropriate error message.

Flag Value	Meaning	Error Message
0	NO ERROR (CLEAR)	—
1	DECIMALS DETECTED	—
2	NOT AN INTEGER	ONLY INTEGERS VALID . . . PLEASE RE-ENTER
3	NOT A NUMBER	ONLY NUMBERS VALID . . . PLEASE RE-ENTER
4	NOT A LETTER	ONLY LETTERS VALID . . . PLEASE RE-ENTER
5	OUT OF LIMITS	OUT OF LIMITS . . . PLEASE RE-ENTER

HANDLING ERRORS

The aim of error handling routines should be to retain program control of input and to restore the input display to the condition it was in before the error occurred. Error messages will make the program 'user friendly' if handled in a sympathetic way. Such messages are best removed once they have been read. The algorithm given in Fig. 4 summarises a suitable method.

Erasure merely means overprinting the display with spaces. There are two cursor resets and there are two ways of doing it: either by printing cursor control characters or alternatively by looking into the operating system and finding out where cursor position bytes are held and poking them back. Of course we have to PEEK into the right location ***before*** we start the input routine at all. So, we should amend Fig. 2 to show a 'FIND CURSOR' box between 'START' and the 'GET CHARACTER $' boxes.

THE MOMENT OF TRUTH

The requirements of data entry are such that the program must be able to cope with idiotic entries such as typing letters when numbers are required, or inadvertantly keying RETURN or other control keys without aborting a program halfway through. Inevitably this involves the checking and validation of all data entry characters, even though this may limit the speed of typing if written in BASIC. In the final analysis an input, once accepted as reasonable by the program and subsequently accepted by the user, although actually incorrect, will be processed as valid data. The error may or may not be significant but garbage in nearly always results in garbage out. Well validated input will help reduce the incidence of garbage input but will ***not*** eliminate it.

SIZE

For a program segment to be commonly used it should be relatively short, especially as the present generation of microcomputers is limited in its free memory capacity, In its smallest available form (Tiny-Mander), the complete routine occupies less than 1K. The heavily commented (Maxi-Mander) listing offered in this article occupies a massive 5.5K and is intended ***only*** as a program for study. The removal of REM statements will reduce it to 1.3K in which form it could be used directly.

PET CHARACTER SET

All the graphics and cursor control characters have been put into the ***Computing Today*** standard format.

PET OPERATING SYSTEM

The 6502 used in the PET allows

zero page addressing, a particularly fast and compact method for storing information, and it is used by Commodore for operating system variables. Zero page merely means the first 256 bytes of memory, which needs only one byte to address. Explanation of this area of the memory map is given in some versions of the **PET User Handbook** and in Nick Hampshire's book, **The PET Revealed**. The New ROM PET uses Page zero much more effectively than the old ROM machines in which many of the variables spill over into Page 2 (Page 1 being used for other things).

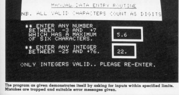
Examination of the memory map soon shows that certain new ROM Page zero addresses hold cursor and keyboard information. These are summarised below:

Old ROM	New ROM	Description
537-538	144-145	Hardware interrupt vector (IRQ)
525	158	No of characters in keyboard buffer
544-545	163-164	Cursor log (row, column)
551	167	Curson on (0 = flashing cursor else off)
549	168	Cursor timing countdown
548	170	Cursor blink flag
224-225	196-197	Pointer to screen line
226	198	Position of cursor on above line (0-79)
245	216	Line where cursor lives

Owners of old ROM machines should use the alternative locations given above, taking care to check that variables take the same values as those given which are for New ROM PETs.

During trials of this routine it was thought that the cursor log in 163-169 would be the best start point and routines using the log worked most of the time. For reasons unknown inputs embedded in loops would unpredictably change the display line spacing, adding an unwanted cursor up after some types of error detection routines. This approach is therefore discarded for the moment. Variables in addresses 216 and 196-197 perform similar functions.

The screen line and cursor line need not carry the same information if cursor control characters are used. The cursor line is the actual line where the cursor lives. It may have been moved away from the screen line by cursor movement controls working relative to the screen line. If the edit keys are disabled, as they are in Mander, then either 198 and 216 or 196-198 inclusive can be used alone but in general it is safer to use the listed subroutine.



```
MANUAL DATA ENTRY ROUTINE
NB. ALL VALID CHARACTERS COUNT AS DIGITS

** ENTER ANY NUMBER
 BETWEEN  -3 AND  +7          5.6
 WHICH HAS A MAXIMUM
 OF SIX CHARACTERS.

** ENTER ANY INTEGER
 BETWEEN -25 AND +76.         22.

ONLY INTEGERS VALID.. PLEASE RE-ENTER.
```

The program as given demonstrates itself by asking for inputs within specified limits. Mistakes are trapped and suitable error messages given.

A major problem is also encountered in switching off the cursor without leaving a pixel block permanently displayed. If the cursor blink is on as the cursor is switched off then the pixel remains displayed. Much effort was devoted to looking into addresses 168 and 170 and trying to ensure the cursor was only switched off when the cursor pixel was in the 'off' part of the blink cycle. This was partially successful but differences always occurred between cursor removeal under program control such as when the maximum length was met and cursor removal by entering RETURN. This method should lead to a solution, but termination of the input routine by printing a space solved the problem very neatly at the expense of making the pro forma 'window' at least one space longer than the specified input length. For presentation aesthetics an extra space at the beginning is also needed. The cursor is best displayed, clearly isolated, in the second character position of the 'window' when inviting data entry.

Addresses 144 and 145 are pointers to interrupt service routines. By changing the contents of the low byte, 144, to 49 (the high byte remaining unchanged) and skipping the STOP key detection routine, the STOP key is effectively disabled. The normal content of 144 is 46. Even when disabled the edit part of STOP key action still occurs and the cursor movement generates an error. This can be avoided by looking into the keyboard Peripheral Interface Adaptor PIA 1 which lives at addresses 59408-11. 59410 is the register address of the keyboard row input byte. If the fifth least significant bit is set to logical zero (which happens when the STOP key is pressed) with all the other bits remaining at logical 1, then the number 239 is generated. PEEK (59410), if equal to 239, detects STOP even when it is disabled. The conditional loop of line 640 thus avoids the edit problem.

One final point: the keyboard as a whole can be ▶

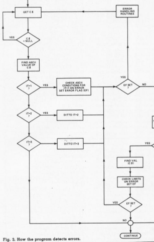


Fig. 3. How the program detects errors.

disabled by setting the least significant bit of PIA 1 register address 59411 (CB 1) to logical zero. This can be done by a POKE 59411, 60. To enable the keyboard POPKE 59411, 61. This option in ***not*** used in Mander.

CRASH PROOF INPUT

This section is divided into four subsections:

560-590 — Set up
600-750 — Enter valid data else re-enter
760-800 — RETURN (exit) routine
830-900 — DELETE (erase)

The setting up uses the Page zero variable addresses to place the cursor, disable the STOP key a POKE 144, 46 must be included at the end of the main program or else a direct POKE must be used after the program has finished. The final way to reset is to switch off and back on. The action of all edit/cursor control keys except DELETE is ignored by the statement in lines 660 and 670. The checks, except for the length of the input string (I$), are handleded in other subroutines but the necessary GOSUBs appear in this section. Condition statements pointing to exit and erasure are embedded in the section 600-760. The code is best followed using the algorithm in Fig. 2.

ALPHANUMERICAL CHECKS

The program lines 920 to 1090 perform the alphanumerical checks. The series of condition statements at the beginning is used for clarity but in Tiny-Mander a single statement of type ON . . .GOTO occupies less space and executes more quickly. For numerical inputs the numbers 0-9, CHR$(48) to (57), the decimal point CHR$(46), the plus CHR$(43) and minus signs CHR$(45) are the only valid characters. The plus and minus signs must lead and only one decimal point can be allowed. When one decimal point has been detected then the error flag (EF) is given the value 1. If a second decimal point is detected then a 'non-numerical' flag is set: EF = 3. For alphabetic inputs the hyphen, CHR$(45), the apostrophe CHR$(44) and of course the full stop, CHR$(46), together with normal letters CHR$(65) to (90), must all be accepted as valid. If an invalid character is detected than EF is set to 4 meaning the character is not a letter.

LIMIT CHECKS

The limit checks are very straightforward. If IV (in its second and final use as the numerical value of the input string) is outside the specified limits (LL & UL) then an 'out of limits' error flag is set: EF = 5. In practice two other conditions must be accomodated:

(a) Once any error is detected and EF set then further checks must be skipped or EF may be reset and generate an incorrect error message.

(b) The statement VAL returns the value zero (0) for alphabetic input as well as for the numerical value zero. The ambiguity can be resolved by assuming that the numerical value of the leading character must also be a zero. The penalty

```
100 REM ** VARIABLE ALLOCATION
110 REM ** VARIABLE DIMENSIONS
120 DIM E$(6):REM ** ERROR STRINGS
130 DIM C$(1):REM ** INPUT CHARACTER
140 DIM I$(1):REM ** INPUT STRING
150 DIM IV(1):REM ** INPUT VARIABLES
160 DIM IT(1):REM ** INPUT TYPE
170 DIM IL(1):REM ** INPUT LENGTH
180 DIM EF(1):REM ** ERROR FLAG
190 DIM CC(1):REM ** CHARACTER COUNT
200 DIM CP(4):REM ** CURSOR POSITION
210 DIM LL(1):REM ** LOWER LIMIT
220 DIM UL(1):REM ** UPPER LIMIT
230 DIM C(1):REM ** COUNT
240 REM ** EXPLANATION
250 REM ** INPUT VARIABLE (IV)
260 REM ** 1ST USE = ASCII OF C$
270 REM ** 2ND USE = VALUE OF I$
280 REM ** INPUT TYPE (IT)
290 REM ** IT=0 ALLOWS ANY CHARACTER
300 REM ** IT=1 ALLOWS ANY FP NUMBER
310 REM ** IT=2 ALLOWS INTEGERS ONLY
320 REM ** IT=3 ALLOWS LETTERS ONLY
330 REM ** ERROR FLAG (EF)
340 REM ** EF=0 NO ERROR
350 REM ** EF=1 ONE DECIMAL POINT
360 REM ** EF=2 NOT AN INTEGER
370 REM ** EF=3 NOT A NUMBER
380 REM ** EF=4 NOT A LETTER
390 REM ** EF=5 OUT OF LIMITS
400 REM ** ERROR MESSAGE STRINGS
410 E$(1)="[19 SPC]"
420 E$(2)="ONLY INTEGERS VALID.."
430 E$(3)="ONLY NUMBERS VALID..."
440 E$(4)="ONLY LETTERS VALID..."
450 E$(5)="OUT OF LIMITS..."
460 E$(6)="[SPC]PLEASE RE-ENTER."
470 GOTO 1500
480 REM ** INPUT SUBROUTINE LIST
490 REM ** 1-FIND CURSOR VARIABLES
500 FOR C=1 TO 3
510 CP(C)=PEEK(195+C):REM ** POINT TO SCREEN LINE
520 NEXT C
530 CP(4)=PEEK(216):REM ** 198,216 = CURSOR POS
540 RETURN
550 REM ** 2-CRASHPROOF INPUT
560 POKE 144,49:REM ** DISABLE STOP KEY
570 GOSUB 1260:REM ** CLEAR LAST INPUT
580 POKE 167,0:REM ** REPLACE CURSOR
590 POKE 158,0:REM ** CLEAR BUFFER
600 GET C$:REM ** GET A CHARACTER
610 IF C$="" THEN 600:REM ** EMPTY
620 IV=ASC(C$):REM ** FIND ASCII CODE
630 REM ** AVOID STOP KEY EDIT ACTION
640 IF PEEK(59410)=239 THEN 600
650 REM ** IGNORE EDIT KEYS
660 IF IV=17 OR IV=18 OR IV=19 OR IV=29 THEN 600
670 IF IV=145 OR IV=147 OR IV=148 OR IV=157 THEN 600
680 IF IV=20 AND CC=0 THEN 850:REM ** RESET
690 IF IV=13 THEN 780:REM ** RETURN
700 IF IV=20 THEN 850:REM ** ERASE LAST CHAR
710 GOSUB 930:REM ** CHARACTER CHECK
720 PRINT C$;:REM ** PRINT CHAR
730 I$=I$+C$:REM ** JOIN CHARACTERS
740 CC=LEN(I$):REM ** CHECK FOR LENGTH
750 IF CC>=IL THEN 780
760 IF EF=0 OR EF=1 THEN 600:REM ** LOOP
770 REM ** INPUT EXIT ROUTINES
780 IF I$="" THEN 600:REM ** LOOP IF EMPTY
790 IF EF=1 THEN EF=0:REM ** NO ERROR
800 POKE 167,1:REM ** REMOVE CURSOR
810 PRINT CHR$(32):REM ** END INPUT
820 IF IT=1 OR IT=2 THEN GOSUB 1120:REM ** LIMIT
830 RETURN
840 REM ** ERASE LAST CHAR
850 IF I$="" THEN 600:REM ** NIL TO ERASE
860 REM ** ADJUST IF DECIMAL POINT
870 IF ASC(RIGHT$(I$,1))=46 THEN EF=0
880 PRINT "[CL][2 SPC][2 CL]";
890 CC=CC-1
900 I$=LEFT$(I$,CC)
910 GOTO 600
920 REM ** 3-ALPHANUMERIC CHECKS
930 IF IT=2 THEN 1010:REM ** INTEGERS
940 IF IT=1 THEN 1030:REM ** NUMBERS
950 IF IT=0 THEN RETURN:REM ** NO CHECKS
960 REM ** ALLOWS ONLY LETTERS ', AND -
970 IF IV<65 AND IV<>32 AND IV<>39 AND IV<>45 AND IV<>46 THEN EF=4
980 IF IV>90 AND IV<193 OR IV>90 AND IV>218 THEN EF=4
990 RETURN
1000 REM ** ALLOWS ONLY INTEGERS
1010 IF IV=46 THEN EF=2
1020 REM ** ALLOWS ONLY ONE DP
1030 IF IV=46 AND EF=1 THEN EF=3
1040 IF IV=46 AND EF=0 THEN EF=1
1050 REM ** ALLOWS ONLY NUMBERS .+ AND -
1060 IF IV<48 AND IV<>43 AND IV<>45 AND IV<>46 OR IV>57 THEN EF=3
1070 REM ** ALLOWS ONLY LEADING + OR -
1080 IF CC<>0 AND IV=43 OR CC<>0 AND IV=45 THEN EF=3
1090 RETURN
1100 REM ** 4-LIMIT CHECK
1110 REM ** SKIP ON ERROR
1120 IF EF<>0 THEN RETURN
1130 IV=VAL(I$)
1140 IF IV<LL OR IV>UL THEN EF=5
1150 REM ** ALLOWS FOR A ZERO (0) INPUT
1160 IF IV=0 AND LEFT$(I$,1)<>CHR$(48) THEN EF=3
1170 RETURN
1180 REM ** 5-ERROR MESSAGE DISPLAY
1190 REM ** RESETS NULL INPUT
1200 IF EF=0 AND I$="" THEN 1260
1210 PRINT "[2 CD]";E$(EF);E$(6)
1220 FOR C=1 TO 1000
1230 NEXT C
1240 PRINT "[CU]";E$(1);E$(1)
1250 REM ** 6-DELETE LAST INPUT
1260 GOSUB 1370:REM ** REPLACE CURSOR
1270 FOR C=1 TO IL:REM ** REPLACE INPUT CHARACTERS
1280 PRINT "[SPC]";
1290 NEXT C
1300 GOSUB 1370:REM ** REPLACE CURSOR
1310 IV=13:REM ** CLEAR DELETE
1320 I$="":REM ** CLEAR STRING
1330 CC=0:REM ** CLEAR CHAR.COUNT
1340 EF=0:REM ** CLEAR ERROR FLAG
1350 RETURN
1360 REM ** 7-REPLACE CURSOR
1370 FOR C=1 TO 3
1380 POKE 195+C,CP(C)
1390 NEXT C
1400 POKE 216,CP(4)
1410 RETURN
1420 REM ** 8-ERROR TRAP INPUT
1430 GOSUB 500:REM ** FIND CURSOR
1440 GOSUB 560:REM ** CRASHPROOF INPUT
1450 REM ** ALLOWS USE OF DELETE
1460 IF EF=0 AND I$<>"" THEN RETURN
1470 GOSUB 1200:REM ** DELETE ANY ERROR
1480 GOTO 1430:REM ** NEXT INPUT LOOP
1490 REM ** MAIN PROGRAM
1500 C$=""
1510 POKE 59468,12
1520 PRINT "[CLS]";
1530 FOR C=1 TO 80
1540 C$=C$+"[REV][SPC][OFF]"
1550 NEXT
1560 FOR C=1 TO 3
1570 PRINT C$;
1580 NEXT
1590 PRINT "[HOM]"
1600 PRINT TAB(8)"[REV]MANUAL DATA ENTRY ROUTINE[OFF]"
1610 PRINT TAB(8)"[CU][REV][25↑■][OFF]"
1620 PRINT "[REV]NB. ALL VALID CHARACTERS COUNT AS DIGITS[OFF]"
1630 PRINT
1640 REM ** DRAW A BOX WITH THICK LINES
1650 PRINT TAB(25)"[↑,][8↑"][↑;]"
1660 PRINT TAB(25)"[REV][↑!][CD][CL][↑!][CD][CL][↑!][OFF]";
1670 PRINT "[8↑"][↑!][CU][CL][↑!][CU][CL][↑!]"
1680 PRINT "[2 CU][SPC]**ENTER ANY NUMBER"
1690 PRINT "[2 SPC]BETWEEN -3 AND +7"
1700 PRINT "[2 SPC]WHICH HAS A MAXIMUM"
1710 PRINT "[2 SPC]OF SIX CHARACTERS."
1720 PRINT "[3 CU]"
1730 PRINT TAB(25)"[REV][↑!][OFF][SPC]";
1740 LL=-3:REM ** LOWER INPUT LIMIT
1750 UL=7:REM ** UPPER INPUT LIMIT
1760 IL=6:REM ** INPUT STRING LENGTH
1770 IT=1:REM ** ALLOWS FP INPUT
1780 GOSUB 1430:REM ** ERROR TRAP INPUT
1790 A=IV:REM ** RE-ASSIGN INPUT VARIABLE
1800 PRINT TAB(25)"[↑,][5↑"][↑;]"
1810 PRINT TAB(25)"[REV][↑!][CD][CL][↑!][CD][CL][↑!][OFF]";
1820 PRINT "[5↑"][↑!][CU][CL][↑!][CU][CL][↑!]"
1830 PRINT "[CU][SPC]**ENTER ANY NUMBER"
1840 PRINT "[2 SPC]BETWEEN -25 AND +76."
1850 PRINT "[2 CU]"
1860 PRINT TAB(25)"[REV][↑!][OFF][SPC]";
1870 LL=-25
1880 UL=76
1890 IL=3
1900 IT=2
1910 GOSUB 1430
1920 B=IV
1930 PRINT TAB(12)"[↑,][25↑"][↑;]"
1940 PRINT TAB(12)"[REV][↑!][CD][CL][↑!][CD][CL][↑!][OFF]";
1950 PRINT "[25↑"][↑!][CU][CL][↑!][CU][CL][↑!]"
1960 PRINT "[CU][SPC]**ENTER"
1970 PRINT "[2 SPC]YOUR NAME."
1980 PRINT "[2 CU]"
1990 PRINT TAB(12)"[REV][↑!][OFF][SPC]";
2000 IL=23
2010 IT=3
2020 GOSUB 1430
2030 N$=I$
2040 FOR C=1 TO 1000
2050 NEXT
2060 PRINT "[CLS][REV][SPC]RESULTS:-[SPC][OFF]"
2070 PRINT "[2 CD]YOUR DATA ENTRIES HAVE BEEN RE-ASSIGNED"
2080 PRINT "[CD]AS FOLLOWS:-"
2090 PRINT "[2 CD][SPC]** THE FULL FP NUMBER IN BOX 1 IS NOW"
2100 PRINT "[CD][5 SPC]IN VARIABLE 'A'.[3 SPC]A=[SPC][REV]";
     A"[CL][SPC][OFF]"
2110 PRINT "[CD][SPC]** THE INTEGER IN BOX 2 IS NOW"
2120 PRINT "[CD][5 SPC]IN VARIABLE 'B'.[3 SPC]B=[SPC][REV]";
     B"[CL][SPC][OFF]"
2130 PRINT "[CD][SPC]** YOUR NAME IS NOW IN VARIABLE 'N$'."
2140 PRINT "[CD]"TAB(5)"N$=[SPC][REV]'[SPC]";N$"[SPC]'[OFF]"
2150 FOR C=1 TO 6000
2160 NEXT
2170 POKE 144,46:REM ** RE-ENABLE STOP KEY
2180 PRINT "[CD][SPC]** TO QUIT PRES 'STOP' KEY NOW[CD]"
2190 F$="[3 SPC]THE NEXT ROUTINE WOULD START HERE"
2200 FOR C=1 TO 5
2210 PRINT F$
2220 GOSUB 2270
2230 PRINT "[CU]";E$(1);E$(1)"[CU]"
2240 GOSUB 2270
2250 NEXT C
2260 GOTO 1500
2270 FOR D=1 TO 500
2280 NEXT D
2290 RETURN
```

```
2150 FOR C=1 TO 6000
2160 NEXT
2170 POKE 144,46 REM ** RE-ENABLE STOP KEY
2180 PRINT "[CD][SPC]** TO QUIT PRES 'STOP' KEY NOW[CD]"
2190 F$="[3 SPC]THE NEXT ROUTINE WOULD START HERE"
2200 FOR C=1 TO 5
2210 PRINT F$
2220 GOSUB 2270
2230 PRINT "[CU]";E$(1);E$(1)"[CU]"
2240 GOSUB 2270
2250 NEXT C
2260 GOTO 1500
2270 FOR D=1 TO 500
2280 NEXT D
2290 RETURN
```

is that entry formats of the type ± 0 will be rejected. A more rigourous condition statement may prove necessary in some cases but in Tiny-Mander a less rigourous condition, requiring a zero character on its own, saves space.

ERROR MESSAGES

These routines, located in program lines 1180-1410, merely implement the algorithm in Fig. 4. The error message is given the same string variable number as the error flag variable. The error strings themselves are defined in lines 410-460. Line 1200 is included to avoid conflict on null inputs and skips the error message section completely. It is a personal preference for error messages to occur as near as possible to the offending input and therefore the messages are printed two lines lower than the input. Others may feel that such messages should appear elsewhere, say at the bottom of the display. This is easily achieved by placing the cursor at the beginning of the chosen line and using a routine based on the lines 490-540 to find the cursor position variables. These new values must be POKEd back prior to printing; the error message will then be written in the new position.

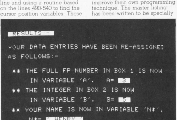
MAXI-MANDER

The main program beginning at line 1490 generates a pro forma window type of display for data entry. Each of the types of input is used in turn and the program demonstrates clearly how to use Mander. After each input the data variables I$ and IV are reassigned and incoporated into a results display which STOP key is enabled. This allows escape from the routine which otherwise would loop back to the beginning.

CONCLUSION

The usefulness of routines such as Mander is self-evident. In practice it might not be necessary to use all of it. Certain parts could perhaps be converted to machine code for faster more compact operation.

The educational nature of the article should have helped beginners to understand how to improve their own programming technique. The master listing has been written to be specially legible and unambiguous. The working version, Tiny-Mander, collapses all variables into arrays I & I$. The specification variables become I1, I2, I3 and I4 instead of IT, IL, LL and UI. Tiny-Mander occupies just under 1K and is just about the maximum size that can be accommodate in a fixed memory. If called from disc storage then this limitation can be overcome.



```
 RESULTS - 

YOUR DATA ENTRIES HAVE BEEN RE-ASSIGNED
AS FOLLOWS:-

 ** THE FULL FP NUMBER IN BOX 1 IS NOW
     IN VARIABLE 'A'.   A=  3 
 ** THE INTEGER IN BOX 2 IS NOW
     IN VARIABLE 'B'.   B=  5 
 ** YOUR NAME IS NOW IN VARIABLE 'N$'.
    N$= ' HENRY '
```

The end of the demonstration routine shows how the checked inputs are transferred to 'normal' variables for use elsewhere.

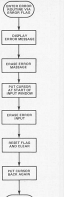


Fig. 4. Error handling after detection.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Games of the unexpected

for the unafraid . . .

THE WHITE BARROWS
Somewhere amid this maze of burial chambers lurks an Evil Sorcerer whom you need to trap. Trouble is, he's protected by Trolls, Dwarves, Serpents and the occasional Dragon or two! Your magic staff will block the tunnel to prevent him escaping unless, that is, he outwits you.
A real brain twister, White Barrows requires both brains and brawn from its players. It's no good just hacking your way through the Barrows and hoping to fall over the Sorcerer. Eventually you'll meet a Dragon, and they don't hack easily! You'll need all your strength and cunning to survive this one for long.
THE WHITE BARROWS Only £6.50 all inclusive!

CELLS AND SERPENTS
More monsters than you ever thought could live behind your keyboard. Wander the hills in search of gold and glory but be very, very careful where you tread! There are things here that will make your wildest nightmares look like Julie Andrews. Fancy meeting a Mind Flyer, for example? Or how about shaking hands with an Asmodeus? (You'll only do that once!) Treasure is here to be found though. . . the hard way.
See just how good you really are at adventuring with this practically unsurvivable fantasy. Not for the faint of heart or the slow of sword.
CELLS AND SERPENTS Only £6.50 all inclusive!

*** SPECIAL DEAL ***
Both programs for only £11.45 all inclusive!

Our Adventure Series programs are available on tape for the following systems:
Commodore VIC-20 (not available for White Barrows), Commodore PET, Sharp MZ-80A and MZ-80K, Tandy TRS-80 Model 1, BBC Model B or 32K Model A, Atari 400 and 800, Sinclair 48K ZX Spectrum.

ASP Software, ASP Ltd,
145 Charing Cross Road, London WC2H OEE

Please send me . . . tape(s) of the following programs:
The White Barrows . . . . @ £6.50 each
Cells and Serpents . . . . @ £6.50 each
Both tapes at special price . . . . only £11.45
My system is a . . . . computer

TRADE ENQUIRIES WELCOME

I am enclosing my Cheque/Postal Order/Money Order (delete as necessary) for £ . . . (payable to ASP Ltd)
OR Debit my Access/Barclaycard (delete as necessary)
☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Please use BLOCK CAPITALS and include your postcode
NAME (Mr/Ms) . . . .
ADDRESS . . . .
. . . .
. . . . POSTCODE . . . .
Signature . . . . Date . . . .

Chris Wyatt

VIC BLOW-UP

If you've ever wondered just how the characters your computer displays are made up then this program will reveal all.

One of the nicest things about Commodore's microcomputers is the ready-made graphics set that comes as standard with their machines. The VIC-20's character set is probably the best within the existing range, so I thought it might be interesting to have a look at exactly how the VIC sticks its pixel dots together to make up what is really a rather friendly set of characters.

THE SHAPE OF THINGS TO COME

As you doubtless know already, the VIC screen display is made up of 23 rows of 22 characters per row, giving a total of 506 character positions in all.

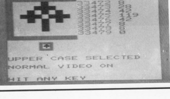
Each of these character positions can itself be divided up into eight rows of eight pixels per row — a pixel being the smallest area of the screen that can be individually controlled. On the VIC these pixels are rectangular, with a width to height ratio of approximately 3 to 2. Since the complete character position is, as we said, built up in eight rows of eight pixels, then this too will be rectangular in the same proportions.

Thus, if you were able to magnify an individual pixel eight times you would find yourself looking at a normal size 'Reverse Space' character.

This technique is used by the program to 'blow-up' any of the characters within the VIC's ROM held character set by magnifying each of the 64 pixels making up the character.

THE PROGRAM TECHNIQUE

The program works by identifying the location in ROM character generator of a character typed in at the keyboard. The eight bytes holding the character 'shape information' are then retrieved from the character generator, and the 'bit-pattern' translated into eight rows of eight full character positions on the VIC's screen display. A 'Reverse Space' character is used to represent a pixel 'set'; a normal 'Space' for a pixel not set.



```
76543210          ADDRS   VAL
                  33472   8
                  33473   28
                  33474   42
                  33475   119
                  33476   42
                  33477   8
                  33478   8
                  33479   0

UPPER CASE SELECTED
NORMAL VIDEO ON

HIT ANY KEY
```

The screen thus displays the exact arrangement of pixels as used in the formation of the normal size character — only eight times bigger. The screen also displays the memory locations of each of the eight bytes used for the character, and the values stored in those locations — both in decimal. All of the VIC character set is accessible, including all upper and lower case alpha-numerics and graphics, plus all 'reverse video' forms. The mode in which the program is currently operating, ie:

1) Upper case/normal video
2) Upper case/reverse video
3) Lower case/normal video
4) Lower case/reverse video

is continuously displayed. Switches between the various modes — upper to lower case, reverse video on and off — are made in the usual ways, which if you're not sure about them, are:

1) Change from upper to lower case and back — press Shift and 'Commodore' keys together.
2) Change to reverse video — press 'CTRL' and 'RVSON' together.
3) Change back to normal video — press 'CTRL' and 'RVSOFF' together.

Invalid entries from the keyboard, eg 'Return', 'INST/DEL', 'function keys', etc, are error-trapped. The program does not crash, but can be interrupted by pressing the 'RUN/STOP' key.

The listing is short enough not to tax even the most weary fingers, and the full *Computing Today* standards have been adhered to without.

If you're into re-defined character generators the program provides a quick and easy method of identifying the memory location of all the VIC's 'off the peg' characters for transferring into your own character generator as required.

Even if you're not a graphics freak, I think you'll find discovering how Commodore designed the VIC's excellent character set an entertaining experience in itself.

PROGRAM NOTES

The program is written in BASIC for the unexpanded VIC-20, but the following notes, together with the accompanying program description and Table 1, should enable the listing to be converted to any micro with a similar type of character generator.

The VIC screen display is controlled by two separate areas of memory. The first, starting at 7680 decimal, is used to contain the character or POKE code, whilst the second, starting at 38400 decimal, contains the code for the colour of the screen position. Thus:

```
POKE 7680,1:POKE 38400,0
```

causes a black A to be displayed at the top left corner of the screen — 1 being the character or POKE code for an A, and 0 being the colour code for Black. Address 36879 controls the background/border colour combination. Address 36874 controls the pitch of the first tone generator of which the VIC has three, plus a white noise generator and address 36878 controls the volume of the sound generators.

Address 36869 is one of the busiest addresses in the VIC, and a full description of everything it does would take up an article in itself. Suffice it to say that in this program, if it contains 240 decimal then the VIC is working in upper-case; if 242 then lower-case has been selected.

Finally, besides POKEing characters onto the screen, the VIC has a particularly powerful PRINT command. The horrendous looking line 290 is a good example of the flexibility of the VIC PRINT statement. The listing uses the ***Computing Today*** standards for representing graphic and cursor control statements, so for those with the stomach for it, this is what line 290 actually does.

The cursor is first HOMEd to the top left of the screen as a starting reference point. It is then moved down 13 rows and across four columns. The cursor colour is set to Green and then ▶

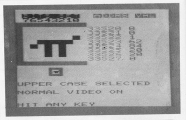
VARIABLES

A$	Single character input by user
CH	POKE code of A$
CS$	Indicates normal or reverse video selected
CA	Indicates upper or lower case selected
CS	Pointer to character generator shape tables
CT	Character video RAM location
CO	Colour video RAM location
I	Counter used to point to each of the eight addresses containing the character shape information.
J	Counter used to point to each of the eight bits in each address containing the character shape information.
VL	Value held in each of the addresses containing the character shape information.
MX	Used as a reference value in conversion of the character generator bit patterns to the 'magnified' display
X	Delay loop counter in error trap routine.

PROGRAM STRUCTURE

Line	Function
Lines 10-20	Set border colour to Blue/background to White and clear screen.
Lines 30-110	Loop displaying current operating mode and testing for mode changes and/or single character inputs by user.
Line 120	PRINT single character input at top left of screen.
Lines 130-140	Get POKE code of character by PEEKing top left of screen; check input is a 'printable' character and if not, branch to error trap at line 310
Line 150	If input is a reverse video character, add 128 to POKE code.
Line 160	Locate start of shape tables for required character; initialize character and colour video RAMs.
Line 170	If a lower case character, redirect to lower case shape tables.
Lines 180-280	MAIN DISPLAY: Print magnified version of character by converting bit patterns from shape tables into patterns of full-size Reverse Spaces. Print the eight addresses containing the shape information and the values, in decimal, held in them.
Line 290	Print the input character at normal size within a Green border.
Line 300	Hand back to keyboard control loop.
Lines 310-360	Error trap — Clear screen, turn whole screen Red, sound audible alarm, advise of error, pause and resume program.

three graphic symbols — a 'bottom right-hand quarter block', a 'bottom half bar', and a 'bottom left-hand quarter block' — are PRINTed. The cursor is now moved down one row and back left three columns, reverse video is turned on, and a reverse 'left-hand bar' symbol (ie a 'right-hand bar') is PRINTed. Normal video is restored, the cursor moved one column to the right, and a 'left-hand bar' symbol PRINTed. Now the cursor goes down one row, back left three columns, and a 'top right-hand quarter block' is PRINTed. Turn reverse video back on, PRINT a reversed 'bottom half bar', revert to normal video and PRINT a 'top left-hand quarter block'. Move the cursor up one row and back left two columns, and then change the cursor colour to Black. Now PRINT the contents of CS$ (RVSON or RVSOFF) and then the character held in A$. Finally, change the cursor colour to Blue. And all on one line . . . Phew!



```
76543210

UPPER CASE SELECTED
NORMAL VIDEO ON

HIT ANY KEY
```

```
10 POKE 36879,30
20 PRINT "[CLS]"
30 PRINT "[HOM][22 CD]HIT ANY KEY";
40 IF CS$="[REV]" THEN PRINT "[HOM][19 CD][REV]REVERSE
   VIDEO ON [OFF]":GOTO 60
50 PRINT "[HOM][19 CD]NORMAL VIDEO ON"
60 IF PEEK(36869)<>CA THEN PRINT "[CLS]";
70 IF PEEK(36869)=242 THEN PRINT "[HOM][17 CD]LOWER
   CASE SELECTED":CA=242:GOTO 90
80 PRINT "[HOM][17 CD]UPPER CASE SELECTED":CA=240
90 GET A$:IF A$="" THEN 30
100 IF A$="[REV]" THEN CS$="[REV]":GOTO 20
110 IF A$="[OFF]" THEN CS$="[OFF]":GOTO 20
120 PRINT "[HOM]";A$
130 IF A$<>"[SPC]" AND PEEK(7680)=32 THEN GOTO 310
140 CH=PEEK(7680)
150 IF CS$="[REV]" THEN CH=CH+128
160 CS=32768+CH*8:CT=7747:CO=38467
170 IF PEEK(36869)=242 THEN CS=34816+CH*8
180 PRINT "[CLS][HOM][CD][CR][REV][CTL BLK]BITS SET
    [OFF][2 SPC][REV][CTL PUR]ADDRS[OFF][SPC][REV]
    [CTL BLU]VAL[OFF]"
190 PRINT "[CR][CTL BLK]76543210":PRINT "[REV][CTL GRN]
    [10 SPC][OFF]"
200 FOR I=0 TO 7
210 VL=PEEK(CS+I):MX=128:CT=CT+22:CO=CO+22
220 FOR J=0 TO 7
230 IF VL<MX THEN 250
240 POKE CT+J,160:POKE CO+J,0:VL=VL-MX
250 MX=MX/2
260 NEXT J
270 PRINT "[REV][CTL GRN][SPC]"TAB(9)"[SPC][OFF]
    [CTL PUR]"CS+I"[CTL BLU][CL]"PEEK(CS+I)
280 NEXT I:PRINT "[REV][CTL GRN][10 SPC][OFF][CTL BLU]"
290 PRINT "[HOM][13 CD][4 CR][CTL GRN][G<D][G<I][G<F]
    [CD][3 CL][REV][G<K][OFF][CR][G<K][CD][3 CL][G<C]
    [REV][G<I][OFF][G<V][CU][2 CL][CTL BLK]"CS$;A$:
    PRINT "[CTL BLU]"
300 GOTO 30
310 PRINT "[CLS]";:POKE 36879,42:POKE 36878,15:PRINT
    "[6 CD][CTL WHT][6 CR]ERROR"
320 PRINT "[3 CD][2 CR]NOT A PRINTABLE":PRINT "[4 CR]
    [CD]CHARACTER[CTL BLU]"
330 POKE 36874,200
340 FOR X=1 TO 2000:NEXT
350 POKE 36874,0
360 GOTO 10
```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Paul Hodgkinson

PROGRAM PROTECTION

This simple method of protecting your programs from being copied may not be the most elaborate but it can certainly cause considerable frustration to the unauthorised user.

Despite recent advances in microcomputer technology, unauthorised copying of programs remains a seemingly insoluble problem. Most existing methods attempt to prevent a return to the command level thus making copying impossible. This is achieved by amending appropriate addresses in the machine by using POKE. Such systems are, of course, ineffective if the program is not run.

This technique, although not a complete solution, has certain attributes which make it very effective. No attempt is made to prevent a copy being made and this can be verified against the original; if the copy is loaded into the PET, it will run and the listing is identical. However, when the user returns at a later date with the copy and attempts to use it, the copy appears corrupted and will not run.

CREATING PROTECTED COPIES

The technique listed here is suitable for use with 16K and 32K PETs.

1) Append an additional statement to the program to be protected. List the program and insert SYS 1000 near the beginning. Remember not to append it to a line which already contains a REMark statement because the entire content of a line following REM is ignored. Decimal 1000 is particularly suitable because it implies a small machine code subroutine resident in the second cassette buffer.

2) Determine the top of BASIC text. At this stage it is necessary to determine how long the program is. This is done by examining the pointer to the start of variables. Variables are stored in the RAM immediately following the BASIC text. Enter:

```
PRINT PEEK(43):PRINT PEEK(42)
```

When Return is pressed two figures will appear:

```
4
104
```

These are the high order byte and the low order byte, expressed in decimal, of the pointer to the start of variables. They must now be converted to a four-digit hexadecimal number. Remember Hex is base 16 and the figures 10 to 15 are substituted by A to F. For example, decimal 255 becomes FF and the example above, therefore, becomes 0468 Hex.

3) Enter the Machine Language Monitor (TIM). TIM is resident in 16K and 32K PETs but a cassette version is required for 8K models. Enter:

```
SYS 1024
```

When Return is pressed the monitor is entered and a display of registers will appear.

4) Display the contents of decimal 1000 (Hex 03E8). After the full stop, type:

```
M 03E8,03EF
```

When Return is pressed, the contents of the bytes from 03E8 Hex to 03EF Hex will be displayed on a single line as two-digit hexadecimal numbers starting with 03E8 Hex. If a '?' appears, the space between the M and the 0 of 03E8 Hex has been omitted. Although this is the standard format for using the monitor it seems designed to mislead! It is not the space that is important, any character can replace the space or comma without affecting the operation of the monitor. The start and end addresses must contain four digits and be in the correct position on the line.

5) Place the 6502 op-code 'Return from subroutine' in decimal 1000 (hexadecimal 03E8). Move the cursor up to the 03E8 Hex in front of the line of bytes previously displayed. Move across to the first two digit number, overtype it with 60 and press Return.

6) Save the program using the Machine Language Monitor (TIM). TIM is used because the start and end address can be specified and it is necessary to save the op-code in 03E8 Hex with the BASIC text. After the full stop, enter:

```
S "TITLE",01,03E8,0468
```

Where 'TITLE' is the program name, 01 is the device number to which the save is made (in this case the first cassette), 03E8 Hex is the start address and the last hexadecimal number (in this case 0468 Hex) is the figure previously calculated to be the top of BASIC text. The familiar 'PRESS PLAY AND RECORD ON TAPE 1' should appear when Return is pressed. Operate the cassette deck as normal. If '?' appears you have probably forgotten the space after the S.

7) Exit from the Machine Language Monitor is done by typing X and pressing Return.

You now have a protected copy. The copy will load in the usual way because LOAD detects the start address placed on the tape header by TIM.

HOW IT WORKS

The protected copy contains the contents of the bytes from 03E8 Hex upwards as well as the BASIC text. When the program is LOADed and RUN, SYS 1000 passes control to decimal 1000 (03E8 Hex). The op-code 60(RTS) immediately passes control back to BASIC and the program RUNs as normal.

If an unauthorised copy is made using SAVE, only the BASIC text from 1024 upwards is recorded — the vital op-code in 03E8 Hex is not included. If the unauthorised copy is loaded to check it is correct, it will run as expected. This is because 03E8 Hex is in a protected area of RAM and is unaffected by command level instructions. The op-code from the original copy will still remain to pass control back to BASIC, providing that the PET has not been switched off or the second cassette buffer used in the interim period.

When the user returns at a later date, the copy will LOAD but any attempt to RUN will cause the monitor to be entered followed by a break or a crash depending on the contents of 03E8 Hex at the time.

CONCLUSION

This technique offers a more subtle approach to program protection and attempts to confuse the copier rather than throw down a blatant challenge to defeat the protection.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

LOW PRICE HIGH QUALITY
SOFTWARE FOR
COMMODORE 32K PET AND 64

PURCHASE AND SALES CONTROL £80 + VAT

Runs both purchase and sales ledgers with optional calculation of VAT from the gross or net amount, analysis by accounting period, 'due for payment' report, totals for net VAT and gross, etc.

INVOICE PRINT £80 + VAT

Prints invoices on your own stationery laid out according to your own instructions. This programme is an optional add-on to be used in conjunction with 'purchase and sales control'.

STOCK CONTROL £60 + VAT

Keeps detailed stock records including stock location, re-order level, quantity on order, cost and selling prices and stock valuation.

NOMINAL LEDGER £60 + VAT

Produces trial balance and up to 20 reports in addition to profit and loss and balance sheet. This programme is intended for use on its own, but it can read files set-up by our purchase and sales control and stock control programmes.

INTEGRATED ACCOUNTING SYSTEMS
FROM £400 + VAT

Write or phone for details and complete software list.

ELECTRONIC AIDS
(TEWKESBURY) LTD

Mythe Crest, The Mythe, Tewkesbury
Gloucestershire GL20 6EB
Tel: 0386 831020/0684 294003

16 K RAM PACKS
FOR VIC 20

To the trade only

PLUS 80 GUARANTEE

To beat any manufacturers price on 16K and 24K RAM packs with full one year guarantee.

This offer applies to a minimum order of 10 units.

Write or phone for details to:

PLUS 80

31-33 LOWER ROAD
HARROW
HA2 0DE

Tel: 01-423 6393

Wendy J Palmer

BIBLIOGRAPHY

A quick perusal of books for your Commodore microcomputer.

When you consider the vast number of Commodore microcomputers that are in use in this country it should become fairly clear that this is not a comprehensive list of the books that have been written about them! Indeed with new books coming out all the time it is unlikely that we could have ever got together a comprehensive list. So what we did was to look at a small selection of the books that were to hand and put together some of the more relevant details about them.

Commodore 64 Computing
Ian Sinclair
Granada Publishing
133 pages; £5.95

This is an introductory guide and reference book for all Commodore 64 users. It covers the setting up and operation of the microcomputer and its many facilities in detail. BASIC syntax is comprehensively summarised with examples to serve as a useful reference for the experienced user and a helpful guide for the less experienced.

PET/ CBM BASIC
Richard Haskell
Pretice-Hall International
154 pages; £9.70

This is a practical manual that goes well beyond theory and shows you how to actually write programs in BASIC. Complete with examples illustrated by photographs from the computer's video screen, the book offers a step-by-step approach to top-down programming that will help you to apply fundamental concepts and program a computer easily and quickly.

The PET Index
Michael A F Ryan
Gower Publishing Company Ltd
194 pages; £12.50

The aim of this book is to provide an index to help the reader locate references to the Commodore PET/CBM microcomputer. It covers 17 different publications (290 issues in all). All early references to the VIC-20 have also been included.

PET Games and Recreations
Mac Oglesby, Len Lindsay and Dorothy Kunkin
Reston Publishing Company Inc.
245 pages; £12.95

The title describes the content of this book which includes plan-ahead games, deductive reasoning games, games of chance, language and counting skills games and recreations (purely for entertainment emphasising the PET's graphics capabilities).

Hands-on BASIC with a PET
Herbert D Peckham
McGraw-Hill Book Company
267 pages; £10.95

This book, a modification of an earlier work by the author, is designed to be useful to anyone who wants to learn how to program the PET in BASIC. There are 10 chapters: The PET computer and BASIC; Getting acquainted with the PET; Introduction to BASIC; Computer arithmetic and program management; Input, output, and simple applications; Decisions, branching, and applications; Looping and functions; Working with collections of information; 'Do-it-yourself' functions and subroutines; and Random numbers and simulations.

PET Personal Computer Guide
Adam Osborne, Jim Strasma and Ellen Strasma
McGraw-Hill Book Company
530 pages; £11.95

This book enables you to learn PET BASIC programming, highlight your programs with graphics, music and other special features, to evaluate and install the latest Commodore peripherals, to use the PET modem and voice synthesizer, and to write and edit drawings and programs with the PET's advanced screen editor.

Learning to use the PET Computer
Garry Marshall
Gower Publishing Company Ltd
87 pages; £4.95

This book provides a simple, down-to-earth, jargon-free introduction to the PET and its software. Many applications of the PET are described, including business, educational and hobby uses. Also, a simple and direct introduction to programming the PET is given.

VIC Graphics
Nick Hampshire
Duckworth
185 pages; £6.95

This book provides the reader with an introduction to programming techniques used to generate graphics displays on a VIC-20. The topics covered include using colour, two dimensional shape plotting, shape scaling and stretching, shape movement, shape rotation, plotting using matrix manipulation and three dimensional shape plotting.

VIC Innovative Computing
Clifford Ramshaw
Melbourne House
151 pages; £6.95

This book provides a brief explanation and the listings for the following: Space flight, Hi-speed maze, High-res draw, Warlock, Dragon's lair, Synthesiser, Ganymede, Hoppy, City bomber, Battleship, Maths, Duck shoot, Grand prix, Rat trap, Earth attack, Bomber attack, Blackjack, Squash, Save the shuttle, Hangman, Alien overun, Invasion, Dumper, Siege, Snakes & ladders, Dungeon, Gold, Nuclear attack, Chess and Assassin.

NEW! ZX SPECTRUM TAPE NOW READY!
NEW! EXPANDED DISC VERSIONS FOR
APPLE, PET AND SHARP!

SOFTWARE FROM
COMPUTING
TODAY

THE VALLEY

© ASP LTD 1982

What are you . . .
Barbarian or Wizard?

Choose your character type carefully . . . Barbarians recover quickly but their magic doesn't come easily. A Wizard? Slow on the draw and slow to mature . . . but live long enough and grow wise enough and your lightning bolts are almost unstoppable . . .

The Valley is a real-time game of adventure and survival. You may choose one of five character types to be your personal 'extension of self' to battle and pit your wits against a number of monsters. Find treasure, fight a Thunder-Lizard in the arid deserts of the Valley, conquer a Kraken in the lakes surrounding the Temples of Y'Nagioth or cauterise a Wraith in the Black Tower. In fact live out the fantasies you've only dared dream about. BUT BEWARE . . . more die than live to tell the tale!

You've read the program (Computing Today — April '82) . . . Now buy the tape. Tape versions (£11.45 each inc P&P and VAT) available for: ZX Spectrum (48K), Atari 400 and 800 (32K), Tandy TRS-80 Model 1 Level 2, BBC Model B, Sharp MZ-80A, Sharp MZ-80K (18K), VIC-20 (with 16K RAM pack) and PET (New ROM, 16K RAM minimum).

Disc versions (£13.95 each inc P&P and VAT) available for: Apple II (DOS 3.3), Sharp MZ-80A, Sharp MZ-80K and PET 8032 (8050 drives).

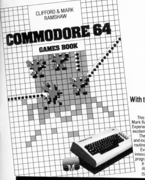
Fill in the coupon and return it to CT Software, ASP Ltd., 145 Charing Cross Road, London WC2H 0EE and become one of the many to play . . . The Valley . . .

Please send me the following versions of The Valley Tape . . . . @£11.45 all inclusive of P&P and VAT.
Disc . . . . @£13.95 all inclusive of P&P and VAT.
I enclose cheque/PO for £ . . . . . (payable to ASP Ltd). OR Debit my Access/Barclaycard (delete as necessary)

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Please use BLOCK CAPITALS
Name (Mr/Mrs/Miss) . . . .
Address . . . .
. . . . Postcode . . . .
Signature . . . . Date . . . .
PS Summer '83

Please allow 21 days for delivery



Teach your Commodore 64 every trick in the book.

With the best software games book ever for your Commodore 64.

This mind blowing collection of game programs, written by Software wizards Clifford and Mark Ramshaw, will turn your Commodore 64 into an entire arcade of electronic fun and thrills. Experience action so fast and furious it will shatter all your concepts about computer game excitement.

The Commodore 64 Games Book is packed with nerve shattering space and adventure games, and includes intriguing strategy games too! Some programs even contain machine language routines.

Every game maximises all the Commodore 64's sophisticated features, including its innovative Sprite capability. And you don't need complex programming skills, because all these programs are very easy to enter.

If you want to know every trick in our book, order your copy today. Book £5.95.
All programs from this book are also available on cassette. Each cassette contains 15 programs. £6.95.

We can teach the VIC 20 a few tricks too!

VIC
INNOVATIVE COMPUTING
Clifford Ramshaw

Vic Innovative Computing

30 highly creative game programs that will open a new dimension of thrills and excitement for every VIC 20 owner.

Featuring arcade favourites like Hoppy, Nuclear Attack, Space Flight and Chess.

The specially designed format ensures that all the programs are very easy to understand and enter.

These programs are also available on a set of 3 cassettes.
Book £6.95.
Each cassette £5.95.

Vic Games Pack

Five fantastic space games on one cassette, including Invaders, Storm and Ground Attack. Every game makes full use of the VIC 20's stunning graphics and sound.

Also included are two 100% machine language programs, Alien Blitz and Space Rocks. This Game Pack is the ultimate test of nerves and skill–amazing value for only £5.95.

Melbourne House
VIC 20
GAMES PACK
5 GREAT GAMES

Melbourne House
VIC 20
The Wizard & the Princess

Wizard and the Princess

In this multi part medieval graphics adventure, you are a bold knight who must rescue the beautiful princess from the grasp of an evil Wizard.

To succeed, you must storm the castle, slay fire breathing dragons and fight the evil troll to the death. A challenge that pushes you and the capabilities of your VIC 20 to the limit.

Cassette for standard VIC 20 £5.95.

MELBOURNE HOUSE PUBLISHERS Orders to: 131 Trafalgar Road, Greenwich, London SE10

Correspondence to: Glebe Cottage,
Station Road, Cheddington,
Leighton Buzzard, BEDS LU7 7NA

Trade enquiries welcome.

☐ Please send me your **free** 48 page catalogue.
Please send me:

Books
VIC 20
☐ VIC Innovative Computing £6.95

COMMODORE 64
☐ Commodore 64 Games Book £5.95

Cassettes
STANDARD VIC 20
☐ VIC Innovative Cassette 1 £5.95
☐ VIC Innovative Cassette 2 £5.95
☐ VIC Innovative Cassette 3 £5.95
☐ VIC Games Pack £5.95
☐ The Wizard & The Princess £5.95

COMMODORE 64
☐ Commodore 64 Super Cassette A £6.95
☐ Commodore 64 Super Cassette B £6.95

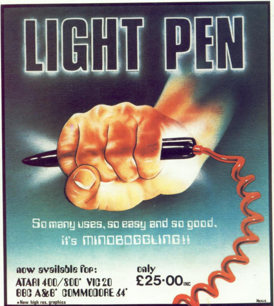
Please add 80p for post and pack £ 80
TOTAL £ ______

All Melbourne House cassette software is unconditionally guaranteed against malfunction.
Access orders can be telephoned through on our 24-hour ansafone (01) 858 7397.

I enclose my cheque/money order for £ ______
Please debit my Access card No. ______ Expiry date ______ Signature ______
Name ______ Address ______
______ Postcode ______

PS6

MELBOURNE HOUSE PUBLISHERS


LIGHT PEN
So many uses, so easy and so good,
it's MINDBOGGLING!!
NEALE.

now available for:

ATARI 400/800* VIC 20
BBC A&B* COMMODORE 64*

*New high res. graphics

only
£25·00 INC

STARTECH
STARTECH
STARTECH
COMPUTERS
sales & services

!!FREE GAME WITH EVERY PEN!!

Please rush me

Name
Address

I enclose Cheque, P/O for
CREDIT CARD ☐ ACCESS ☐ BARCLAY CARD ☐

24 HOUR ANSAPHONE SERVICE

208 Aigburth Rd, Liverpool L17. **051 727-7267**